UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Desenvolupament d'un repositori d'APIs

Treball de final de grau

Autor:

Jaume Lladó Avellaneda

Director:

Carles Farré Tost

Codirectora:

Cristina Gómez Seoane

22/06/2017

Grau en Enginyeria Informàtica

Enginyeria del Software

Resum

Les APIs han experimentat un enorme creixement en els últims anys i estan modificant la forma mitjançant la qual les empreses mostren les seves dades i funcionalitats a l'exterior. La creació, documentació i evolució d'aquestes APIs se segueix fent, però, a mida, amb poc suport per a l'automatització i diverses deficiències conegudes. Amb aquest projecte es pretén desenvolupar un repositori d'APIs amb les funcionalitats necessàries per mostrar i mantenir informació completa i acurada, incloent-hi requisits no funcionals, de les API que s'hi registrin.

Resumen

Las APIs han experimentado un enorme crecimiento en los últimos años y están modificando la forma con la que las empresas muestran sus datos y funcionalidades al exterior. La creación, documentación y evolución de estas APIs se sigue haciendo, aún así, a medida, con poco soporte para la automatización y con diversas deficiencias conocidas. Con este proyecto se pretende desarrollar un repositorio para APIs con las funcionalidades necesarias para mostrar y mantener información completa y acurada, requisitos no funcionales incluidos, de las APIs que se registren en él.

Abstract

Application Programming Interfaces (API) have recently experienced an unexpected growth and are shaping the way in which organizations expose data and functionalities to the outside. The creation, documentation and evolution of these APIs are still done on ad-hoc basis, with little automated support and reported deficiencies. The aim of this project is to develop an API repository with functionalities to maintain a complete and accurate documentation (including non-functional requirements) that will facilitate the adoption of the APIs by third parties.

Índex

Índex de figures

Índex de taules

1. Context del projecte

1.1 Introducció

Una API [1] és un conjunt de funcions i especificacions que les aplicacions informàtiques poden fer servir per a comunicar-se entre elles, de forma que aquestes aplicacions puguin fer servir funcionalitats existents en d'altres, permetent als desenvolupadors estalviar-se haver d'implementar de zero funcionalitats que ja existeixen i poden ser fàcilment aprofitades.

S'entén per repositori aquell lloc on s'emmagatzema i es manté informació digital. En el cas dels repositoris d'APIs, es tracta de webs que es dediquen a mantenir informació sobre APIs com les seves URL, les funcionalitats que tenen implementades per a l'ús dels desenvolupadors o el protocol amb el qual han estat implementades, entre d'altres. Aquests repositoris són d'alta utilitat, ja que permeten trobar diverses APIs dins la mateixa pàgina segons criteris de cerca, i fins i tot, en alguns casos, comparar APIs que ofereixen funcionalitats similars.

Les APIs han experimentat un enorme creixement recentment i estan modificant la forma mitjançant la qual les empreses mostren les seves dades i funcionalitats a l'exterior. La creació, documentació i evolució d'aquestes APIs se segueix fent, però, a mida, amb poc suport per a l'automatització i diverses deficiències conegudes. Aquests desavantatges acaben afectant negativament la productivitat dels desenvolupadors i tenen un impacte negatiu en el temps necessari per llançar a mercat aquestes APIs i tots els serveis que es puguin desenvolupar a partir d'elles.

L'objectiu d'aquest projecte és desenvolupar un repositori d'APIs amb les funcionalitats necessàries per mostrar i mantenir informació completa i acurada (incloent-hi NFR [2]) de les APIs que s'hi registrin, amb la idea de facilitar l'adopció d'aquestes APIs per part dels clients potencials.

El resultat del projecte es pot consultar accedint a aquesta pàgina web, i es pot veure el repositori amb tot el codi desenvolupat en aquest enllaç.

1.2 Stakeholders

A continuació es defineixen les parts interessades en el projecte o stakeholders, és a dir, aquelles que tenen alguna relació amb el projecte, ja sigui per la seva implicació en el desenvolupament o pel seu ús un cop acabat.

1.2.1 Desenvolupador

Aquest és el rol que està implicat en totes les fases del projecte: planificació, especificació, disseny, implementació, testing i documentació. Com que el projecte és un Treball Final de Grau, el desenvolupador és una de les parts que té més interès en l'èxit del projecte, ja que li permet aprovar la carrera un cop l'acabi satisfactòriament.

1.2.2 Directors del projecte

Els directors del projecte són Carles Farré i Cristina Gómez, professors del departament d'Enginyeria de Serveis i Sistemes d'informació (ESSI) de la UPC. Ambdós s'encarreguen d'acordar els requisits de l'aplicació a desenvolupar, així com del seguiment del desenvolupament, i del projecte en general. Ells estan interessats en l'èxit del projecte, ja que formen part del grup de recerca del seu departament (GESSI) i participen en el projecte GENESIS, del qual aquest TFG és una part.

1.2.3 Usuaris finals

Aquest és el grup format pels enginyers de software, o gent relacionada amb aquest àmbit de la informàtica que estan interessats en tenir una eina on poder consultar informació sobre APIs, i on alhora poder presentar aquesta mateixa informació sobre les seves pròpies APIs a la resta d'usuaris.

1.3 Estat de l'art

Per a l'estat de l'art es compta amb un llistat de 15 repositoris [3] proposat pels directors del projecte. Concretament, s'estudien les seves funcionalitats i la informació que proporcionen sobre cadascuna de les APIs que indexen.

Després d'una primera visita als repositoris i un cop vist el que ofereix cadascun d'ells, n'hi ha 10 que queden descartats per les raons següents:

- PublicAPIS: La pàgina ha deixat d'estar disponible.
- Google APIs Discovery Service: Massa limitat, només conté informació sobre les APIs de Google
- API Food: Només redirigeix a les pàgines de cada API, no en mostra informació
- Azure Marketplace: Mercat d'aplicacions per a servidors Azure, no és un repositori d'APIs.
- API Hound: Només redirigeix a les pàgines de cada API, no en mostra informació
- API For That: Només mostra una descripció de les APIs i links les seves respectives homepages.

- Product Hunt: No proporciona cap tipus d'informació sobre APIs, només en té algunes barrejades entre la resta de productes que mostra.
- APIs.io: Tot i que té punts positius com un validador de fitxers api.json (així com un generador dels mateixos), no proporciona cap tipus d'informació sobre les APIs que té. Només redirigeix a les pàgines de les APIs.
- Mashery API Network: Tot i que ofereix eines interessants com auditories per certificar 'developer experience' en les APIs que indexa o eienes de generació de documentació interactiva o SDK, pel que fa a informació pròpiament sobre l'API no ofereix res, només redirigeix a la homepage de l'API.
- APIs.guru: Descartat pel fet que una altra pàgina (API Harmony) mostra totes les API que apareixen en aquesta, i a més, ofereix més funcionalitats i en mostra més informació.

Un cop descartats aquests 10 repositoris, els 5 restants compten amb les característiques següents (resumides a l'annex del document en dues taules):

1.3.1 ProgrammableWeb

ProgrammableWeb [4] és, possiblement, el referent actual pel que fa a repositoris d'APIs per la gran quantitat de funcionalitats diferents que ofereix i el detall amb què compta tota la informació sobre APIs que s'hi pot trobar.

Entre les funcionalitats d'aquest repositori trobem apartats de notícies, tutorials, perfils tant per a APIs com per a usuaris que s'hi registrin, catàlegs d'APIs, SDKs, llibreries, frameworks, web apps i apps mòbils (amb la possibilitat d'afegir qualsevol d'aquests sis elements com a usuari registrat), cercador amb sistema de filtratge, sistema de "tracking" d'APIs per tal que els usuaris puguin mostrar quines són les seves APIs preferides i newsletter.

Pel que fa a informació, ProgrammableWeb ofereix el portal de l'API, un llistat dels seus endpoints, la categoria a la qual pertany l'API, un link a la seva documentació, l'arquitectura amb la qual ha estat desenvolupada, els formats de request i response que suporta, els SDK que s'han creat per a consumir d'aquesta API, tutorials, exemples, llibreries relacionades, detalls sobre els desenvolupadors i un apartat de caràcter social amb gent que ha fet "track" de l'API (seguidors) i comentaris que aquests poden publicar a cada perfil d'API.

Tot i la completesa d'aquest repositori, cal dir que no ofereix cap detall pel que fa a requisits no funcionals.

1.3.2 Exicon

Exicon [5] ofereix funcionalitats com perfil per a les APIs i per als usuaris, la possibilitat d'afegir APIs noves i un cercador.

Respecte a informació sobre APIs, en dóna el portal, la categoria, detalls sobre els desenvolupadors, el protocol que fan servir, els formats de dades que suporten, i el preu que tenen, en cas que en tinguin.

Com en el cas anterior, aquest repositori tampoc té en compte cap requisit no funcional.

1.3.3 Algorithmia

Algorithmia [6] té la peculiaritat que se centra a mostrar APIs que implementen diversos tipus d'algorismes, a part de ser l'únic dels repositoris estudiats inicialment que té en compte algun requisit no funcional.

Pel que fa a funcionalitats, el repositori compta amb perfils per a APIs, perfil per als usuaris que s'hi registrin, un sistema de categorització i ordenació d'APIs, i la possibilitat de fer crides a les APIs que indexa des de la mateixa web, per a poder provar-les fàcilment.

La informació sobre cada API que mostra el repositori consisteix en un apartat per a la descripció de l'API, on els desenvolupadors introdueixen el portal de l'API i s'inclouen exemples que després es poden provar des de la mateixa web, així com exemples de com fer les crides en diversos llenguatges de programació, un altre apartat on es mostra el cost de fer crides a cada API, i un últim on hi ha els permisos que necessita l'API per a les seves crides (si necessita accés a Internet o fer crides a altres APIs, per exemple).

Com deia, aquest repositori és l'únic dels cinc en els quals m'he centrat que té en compte algun requisit no funcional. Concretament, fa servir el temps mitjà de resposta per a calcular el cost de les crides a les APIs que mostra.

1.3.4 API Harmony

API Harmony [7] és el directori d'APIs de l'empresa d'informàtica IBM, i es diferencia lleugerament de la resta de repositoris per la informació que mostra sobre les APIs que indexa.

Les funcionalitats que ofereix són similars a les de la resta de repositoris. Compta amb un blog, un cercador, perfil per a les APIs, classificació d'APIs per més usades, més noves i més populars, i un sistema per a afegir noves APIs al directori.

Pel que fa a informació, aquest és l'únic repositori que ofereix la possibilitat de descarregar l'especificació OpenAPI [8] de les APIs que té, a més d'oferir-ne el portal, diversos endpoints, l'enllaç a la documentació de l'API i la possibilitat de provar l'API des de la seva mateixa web.

També ofereix altres dades interessants com són les APIs que estan relacionades amb la que s'estigui consultant, el top 10 de preguntes a StackOverflow [9] relacionades amb l'API, el nombre d'usos d'aquella API per projectes allotjats en repositoris a GitHub [10] i algunes de les llibreries npm que interactuen amb l'API en qüestió.

Tot i això, aquest directori no té en compte cap dels requisits no funcionals de les APIs que indexa.

1.3.5 RapidAPI

Entre les funcionalitats que RapidAPI [11] ofereix podem trobar un cercador amb classificació d'APIs per categories, perfil per a APIs i per a usuaris, un blog, la possibilitat d'afegir noves APIs al directori, un apartat amb les APIs més populars i més utilitzades, i la possibilitat de provar les APIs des dels seus perfils.

Pel que fa a la informació que mostra sobre les APIs, hi ha els diversos endpoints amb les seves respectives descripcions, la data de l'última modificació de l'API, i el nombre d'instal·lacions.

Com passa en la majoria de casos, aquest repositori tampoc inclou cap requisit no funcional en la informació que proporciona sobre les APIs.

2. Abast del projecte

El projecte consta de dues fases. La primera consisteix en estudiar un subconjunt dels repositoris d'APIs existents triat pels directors del projecte, per tal de veure quines funcionalitats ofereixen i quina informació sobre APIs proporcionen, i decidir, juntament amb els directors, quines d'aquestes funcionalitats i quanta informació es vol mostrar a la plataforma que es desenvoluparà.

Un cop això quedi clar, es segueix amb la segona fase, que consisteix en la implementació d'un prototipus de directori d'APIs, que no només compti amb les característiques vistes en els repositoris consultats a l'apartat d'estudi de l'art, sinó on també hi hagi un tractament dels requisits no funcionals.

2.1 Possibles obstacles

2.1.1 Mal plantejament

Qualsevol problema que sorgeixi o mala decisió que es prengui durant la fase de plantejament del projecte pot tenir conseqüències negatives importants sobre el desenvolupament d'aquest.

És per això que és important fer una bona recerca de l'apartat d'estat de l'art, per tal d'incloure únicament aquelles característiques vistes durant la recerca que siguin realment adequades per al projecte.

2.1.2 Limitació temporal

Un dels aspectes més importants a l'hora de realitzar un projecte és la forma en què s'organitza el temps de què es disposa, que en el cas d'aquest projecte és d'uns quatre mesos, de forma que en el cas que aparegui alguna complicació en el projecte, aquesta haurà de ser resolta amb el mínim temps possible, ja que el marge amb el qual es compta és limitat.

2.2 Metodologia de treball

Com es comenta anteriorment a l'apartat sobre l'abast del projecte, aquest s'estructura en dues fases.

La primera és de recerca, seguint una metodologia de cerca, anàlisi i síntesi que ha de servir per a conèixer l'estat de l'art pel que fa als directoris d'APIs existents i per a tenir un punt de partida de cara a definir els requisits que acaba tenint el sistema desenvolupat.

La segona fase és pròpiament de desenvolupament, i en aquesta se segueix una metodologia àgil adaptada a les circumstàncies del projecte: treball individual amb generació d'una memòria on s'explica tot el procés seguit i els resultats obtinguts.

2.2.1 Eines de seguiment

Per al seguiment de la feina feta, es fa servir principalment correu electrònic per tal de comunicar la finalització de les tasques assignades, i durant la fase de desenvolupament, s'usa també un repositori a Bitbucket [12] on els directors del projecte poden veure el codi que es va produint.

2.2.2 Mètodes de validació

El principal mètode de validació són les reunions periòdiques amb els dos directors, que a més també són els clients del producte a desenvolupar, i que, per tant, poden validar l'exactitud amb la qual s'està desenvolupant el que ells volen que faci en cada iteració.

3. Planificació temporal

3.1 Calendari

El projecte té una durada aproximada de gairebé cinc mesos, tenint en compte que les primeres reunions i el principi de la recerca sobre l'estat de l'art va començar cap a principi del mes de febrer del 2017 i que acabarà cap a finals de juny del mateix any.

Tot i això, és probable que, en algun moment, el calendari real sigui diferent del que es planifica a continuació degut a canvis que es puguin produir en l'ordre o la duració de qualsevol de les tasques que hi apareixen.

3.2 Descripció de les tasques

3.2.1 Planificació del projecte

Aquesta és una de les fases més importants del projecte, ja que és aquí on queden definits molts dels aspectes bàsics del projecte, com són l'abast, els seus objectius, i la planificació temporal.

3.2.2 Estat de l'art

Aquí és on es realitza una recerca sobre l'estat de l'art, important per tal de veure les opcions que ofereixen els repositoris d'APIs existents i poder decidir posteriorment quines d'aquestes interessen de cara a ser implementades en el repositori del projecte.

Aquesta fase té una durada d'aproximadament quatre setmanes, i es realitza de forma paral·lela a la planificació del projecte.

3.2.3 Anàlisi de requisits

Aquesta fase és la primera que no forma part de la planificació del sistema.

Aquí és on es defineixen tots els requisits del sistema, tant els funcionals com els no funcionals. S'espera que aquesta fase tingui una durada d'aproximadament 10 dies.

3.2.4 Especificació del sistema

Aquesta fase també és molt important, ja que és on es decideixen totes les característiques que té el software a desenvolupar a partir de les conclusions que s'extreuen de l'estat de l'art.

Entre aquestes característiques hi ha, principalment, les funcionalitats a implementar i la informació a oferir sobre les APIs que hi ha al repositori.

Per tal de fer tot això, se subdivideix aquesta fase en tasques com l'elaboració del diagrama de classes, l'especificació dels casos d'ús, i en cas que sigui necessari, l'elaboració de diagrames de seqüència corresponents a aquests casos d'ús.

3.2.5 Disseny del sistema

La fase que segueix a l'especificació del sistema és la del disseny. És aquí on es tria l'arquitectura a fer servir en el sistema i on es dissenya pròpiament com serà el sistema.

En concret, es dissenya el sistema externament, definint com són les pantalles de l'aplicació que els usuaris veuen quan hi accedeixen i la navegabilitat entre elles; i internament, explicant les decisions sobre l'arquitectura que es fa servir i el model de les entitats que hi ha al sistema.

3.2.6 Implementació del software

Després d'haver definit com serà el repositori, ja es disposarà de tot el necessari per a començar a implementar-lo.

Per a la implementació se seguiran les pautes que descriuen les metodologies àgils, és a dir, iteracions curtes en les quals s'afegiran noves funcionalitats a les ja disponibles en el software.

En aquestes iteracions s'hi inclouran proves a les funcionalitats afegides, de forma que en cas que hi hagi errors es puguin detectar el més abans possible, contràriament al que passaria si es deixés la totalitat de la fase de proves pel final de la implementació.

També s'inclou en aquesta fase una petita part dedicada a la preparació de l'entorn de treball, ja sigui instal·lació de software, tria de framework [13] de desenvolupament, instal·lació de base de dades, etc.

3.2.7 Proves

Encara que s'hagin provat les funcionalitats durant les iteracions en les quals s'hagin afegit, es deixarà un període de temps al final de la implementació per tal de comprovar que el conjunt de funcionalitats implementades funciona correctament.

3.2.8 Preparació de la defensa

Un cop el repositori estigui acabat i totes les seves funcionalitats hagin estat provades, només quedarà compilar tota la documentació generada durant el desenvolupament del projecte per tal d'incloure-la a la memòria final del projecte, juntament amb la documentació generada a l'assignatura de GEP, i preparar la defensa del projecte davant el tribunal.

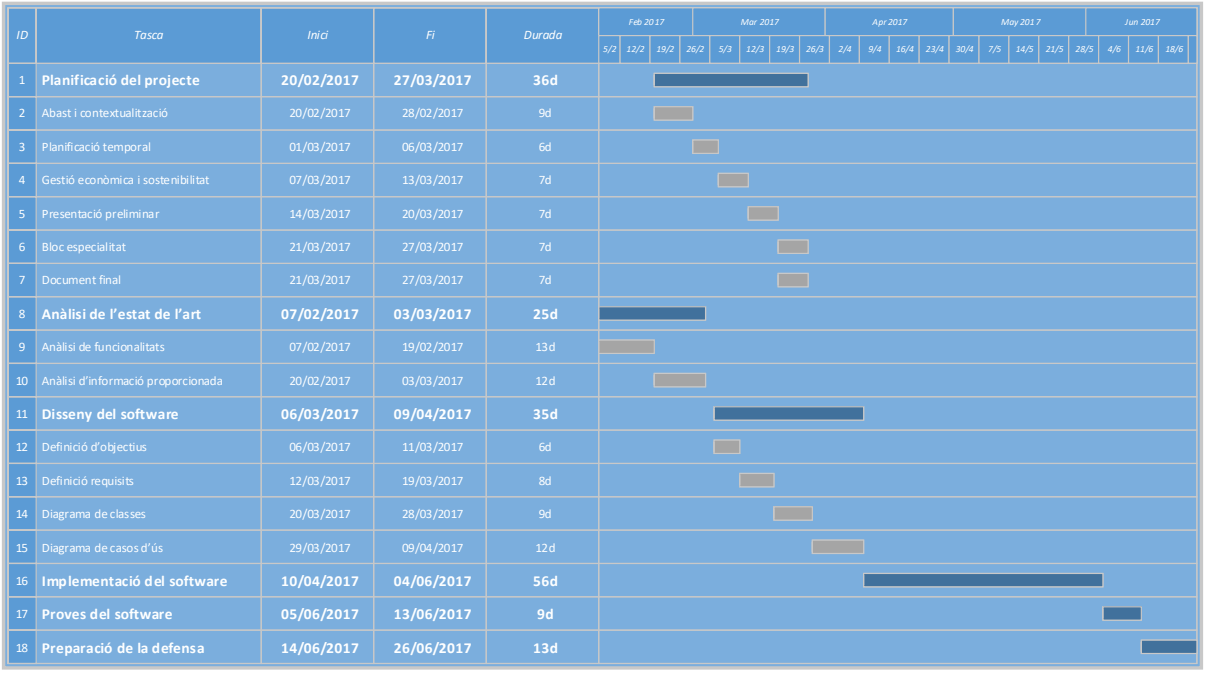
3.3 Estimació temporal

Fase	Temps dedicat (hores)
Planificació	75
Estat de l'art	20
Disseny	60
Implementació	150
Proves	30
Preparació de la defensa	40
TOTAL	375

Taula 1: Estimació temporal

3.4 Diagrama de Gantt

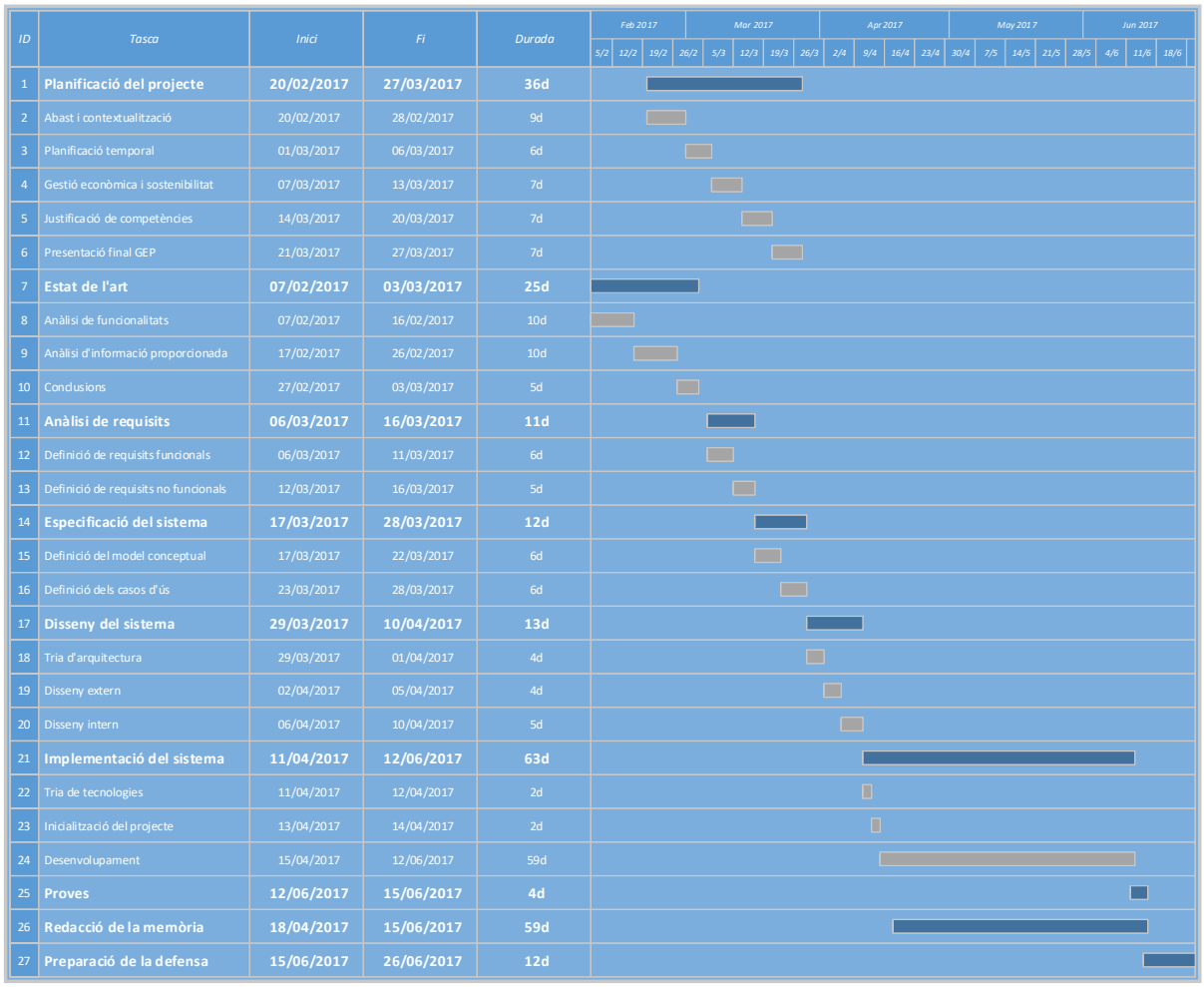
Al primer diagrama es pot veure el diagrama de Gantt corresponent a la planificació temporal feta durant l'assignatura de Gestió de Projectes, prèvia a la fase d'anàlisi de requisits del projecte.



ID	Tasca	Inici	Fi	Durada
1	**Planificació del projecte**	**20/02/2017**	**27/03/2017**	**36d**
2	Abast i contextualització	20/02/2017	28/02/2017	9d
3	Planificació temporal	01/03/2017	06/03/2017	6d
4	Gestió econòmica i sostenibilitat	07/03/2017	13/03/2017	7d
5	Presentació preliminar	14/03/2017	20/03/2017	7d
6	Bloc especialitat	21/03/2017	27/03/2017	7d
7	Document final	21/03/2017	27/03/2017	7d
8	**Anàlisi de l'estat de l'art**	**07/02/2017**	**03/03/2017**	**25d**
9	Anàlisi de funcionalitats	07/02/2017	19/02/2017	13d
10	Anàlisi d'informació proporcionada	20/02/2017	03/03/2017	12d
11	**Disseny del software**	**06/03/2017**	**09/04/2017**	**35d**
12	Definició d'objectius	06/03/2017	11/03/2017	6d
13	Definició requisits	12/03/2017	19/03/2017	8d
14	Diagrama de classes	20/03/2017	28/03/2017	9d
15	Diagrama de casos d'ús	29/03/2017	09/04/2017	12d
16	**Implementació del software**	**10/04/2017**	**04/06/2017**	**56d**
17	**Proves del software**	**05/06/2017**	**13/06/2017**	**9d**
18	**Preparació de la defensa**	**14/06/2017**	**26/06/2017**	**13d**

Figura 1: Planificació inicial

A continuació es mostra el diagrama de Gantt corresponent a la durada real del projecte. Les diferències observables del segon diagrama respecte al primer són el canvi de nom de les subtasques de la fase de Planificació del projecte (corresponent a l'assignatura de Gestió de Projectes), la major granularitat de la fase d'Estat de l'art, la separació de la fase de Disseny en tres fases, anàlisi de requisits, especificació del sistema i disseny del sistema, amb les seves respectives subtasques, la major granularitat de la fase d'implementació i l'addició de la fase de redacció de la memòria.



ID	Tasca	Inici	Fi	Durada
1	**Planificació del projecte**	**20/02/2017**	**27/03/2017**	**36d**
2	Abast i contextualització	20/02/2017	28/02/2017	9d
3	Planificació temporal	01/03/2017	06/03/2017	6d
4	Gestió econòmica i sostenibilitat	07/03/2017	13/03/2017	7d
5	Justificació de competències	14/03/2017	20/03/2017	7d
6	Presentació final GEP	21/03/2017	27/03/2017	7d
7	**Estat de l'art**	**07/02/2017**	**03/03/2017**	**25d**
8	Anàlisi de funcionalitats	07/02/2017	16/02/2017	10d
9	Anàlisi d'informació proporcionada	17/02/2017	26/02/2017	10d
10	Conclusions	27/02/2017	03/03/2017	5d
11	**Anàlisi de requisits**	**06/03/2017**	**16/03/2017**	**11d**
12	Definició de requisits funcionals	06/03/2017	11/03/2017	6d
13	Definició de requisits no funcionals	12/03/2017	16/03/2017	5d
14	**Especificació del sistema**	**17/03/2017**	**28/03/2017**	**12d**
15	Definició del model conceptual	17/03/2017	22/03/2017	6d
16	Definició dels casos d'ús	23/03/2017	28/03/2017	6d
17	**Disseny del sistema**	**29/03/2017**	**10/04/2017**	**13d**
18	Tria d'arquitectura	29/03/2017	01/04/2017	4d
19	Disseny extern	02/04/2017	05/04/2017	4d
20	Disseny intern	06/04/2017	10/04/2017	5d
21	**Implementació del sistema**	**11/04/2017**	**12/06/2017**	**63d**
22	Tria de tecnologies	11/04/2017	12/04/2017	2d
23	Inicialització del projecte	13/04/2017	14/04/2017	2d
24	Desenvolupament	15/04/2017	12/06/2017	59d
25	**Proves**	**12/06/2017**	**15/06/2017**	**4d**
26	**Redacció de la memòria**	**18/04/2017**	**15/06/2017**	**59d**
27	**Preparació de la defensa**	**15/06/2017**	**26/06/2017**	**12d**

Figura 2: Planificació revisada

3.5 Valoració d'alternatives i pla d'acció

Com que aquest projecte es durà a terme amb una adaptació de metodologies àgils (possiblement SCRUM [14]), encara que hi hagi problemes durant el transcurs del projecte, la planificació podrà ser modificada lleugerament per tal de corregir aquestes desviacions.

Per altra banda, també cal dir que si qualsevol de les fases a dur a terme s'acaba en menys temps que s'ha estimat, es passarà a la següent fase per tal d'aprofitar aquestes hores sobrants.

Durant totes les fases se seguiran duent a terme reunions periòdiques amb els directors, de forma que hi haurà un seguiment molt acurat de cadascuna d'elles.

En la taula d'estimació temporal s'han repartit les hores de durada del TFG segons l'equivalència en crèdits ECTS per la qual cada crèdit ECTS equival a 25 hores.

Com que el TFG té 15 ECTS, obtenim un total de 375 hores.

4. Gestió econòmica

4.1 Identificació i estimació de costos

Els costos del projecte es poden dividir en recursos humans i recursos no humans, que seran els aspectes que tindrà en compte el pressupost del projecte.

Concretament, dins de l'apartat de recursos no humans tractarem únicament els recursos hardware, ja que tot el software que es farà servir per a la realització del projecte serà software lliure.

4.1.1 Recursos humans

El projecte serà desenvolupat únicament per una persona, que serà, per tant, l'encarregada de dur a terme els rols tant de project manager i d'analista de software per a la fase de planificació, com d'enginyer de software per a les parts d'implementació i testing.

Segons les fonts consultades [15], els salaris dels dos rols rellevants per al projecte, tenint en compte 250 dies laborables a l'any i una jornada de 8 hores diàries, són els següents:

- Project manager: 40000 €/any → 20,50 €/h
- Analista de software: 29000 €/any → 14,50 €/h
- Enginyer de software: 31000 €/any → 15,50 €/h
- Tester: 32000 €/any → 16,00 €/h

Amb aquestes dades es pot elaborar la taula següent, on es pot veure el cost de cadascuna de les fases del projecte segons el rol que la durà a terme:

Fase	Temps dedicat (hores)				Cost estimat (€)
	Project Manager	Analista	Enginyer de software	Tester	
Planificació	75	0	0	0	1537,50
Estat de l'art	0	20	0	0	290
Disseny	0	0	60	0	930
Implementació	0	0	150	0	2325,00
Testing	0	0	0	30	480
Preparació de la defensa	40	0	0	0	820
Total	**115**	**20**	**210**	**30**	**6382,50**

Taula 2: Estimació de costos per fase

De la mateixa forma, es pot incloure també la taula amb el cost per cadascun dels rols:

Rol	Temps dedicat (h)	Preu per hora (€)	Cost estimat (€)
Project Manager	115	20,5	2357,50
Analista	20	14,5	290.00
Enginyer de software	210	15,5	3255,00
Tester	30	16	480.00
Total	**375**		**6382,50**

Taula 3: Estimació de costos per rol

4.1.2 Recursos no humans

De forma contrària al cas dels recursos humans, aquí s'ha de tenir en compte el temps de vida dels diversos elements, i, per tant, l'amortització que se'n farà.

De cara a totes les fases del projecte faran falta una sèrie de dispositius hardware.

Si tenim en compte la vida útil dels diferents elements i que el temps d'amortització són 6 mesos, podem calcular els costos següents:

Producte	Preu (€)	Vida útil (anys)	Amortització (€)
Acer Aspire E 15	670	5	67
Monitor Dell ST2310	230	10	11,5
Teclat + mouse Logitech	13	5	1,3
Total	**913**		**79,8**

Taula 4: Estimació de costos de recursos no humans

4.1.3 Despeses indirectes

En un projecte de característiques diferents també hi hauria costos d'oficina a tenir en compte, com poden ser aigua, llum o internet, però en el cas d'un treball final de grau, el projecte es realitzarà o en una casa particular o en instal·lacions de la UPC, i per tant, no suposarà cap cost extra.

4.1.4 Contingència

Es reservarà una part del pressupost per a la partida de contingència, concretament un 10% de la suma de costos, de cara a cobrir els imprevistos que hi pugui haver durant la realització del projecte.

4.1.5 Pressupost total

Un cop calculats els costos dels recursos humans i no humans, es poden unir les despeses del projecte en una sola taula:

Tipus	Cost
Recursos humans	6382,50
Recursos no humans	79,8
Contingència	646,23
Total	**7108,53**

Taula 5: Estimació de cost total

Com a consideracions finals, caldrà tenir en compte que no es té present en cap moment un marge de benefici sobre el cost total del projecte, ja que es tracta d'un projecte sense ànim de lucre i no destinat a la venda de cap producte.

A més, també s'hauria de tenir en compte que hi pot haver un cost de manteniment del sistema un cop estigui finalitzat, però ara per ara no se'n pot calcular el cost exacte, ja que dependrà del que s'hagi d'implementar sobre el sistema existent i el que cobri un enginyer de software en el moment que toqui implementar-ho.

5. Sostenibilitat i compromís social

A continuació es presenta el breu estudi realitzat sobre la sostenibilitat de la part de planificació del projecte, segons la matriu de sostenibilitat presentada a la guia de l'assignatura de GEP.

Dimensió	Descripció	Valoració
Econòmica	Viabilitat econòmica	7
Social	Millora de la qualitat de vida	6
Ambiental	Anàlisi de recursos	7
Total		**20**

Taula 6: Valoració de sostenibilitat

5.1 Dimensió econòmica

S'ha realitzat una avaluació dels costos del projecte, tant de recursos humans com de no humans, tenint en compte possibles desviacions durant la realització del projecte.

Encara que el preu del projecte no el faria viable si hagués de ser competitiu (no és l'objectiu del projecte, ja que no es posarà a la venda), el seu cost real és baix a causa de la seva naturalesa acadèmica i pel fet que forma part d'un programa més ampli de recerca del Grup de Recerca d'Enginyeria del Software de la UPC.

En qüestió de sostenibilitat econòmica, el projecte mereix una valoració de 7.

5.2 Dimensió social

L'objectiu del projecte és oferir als usuaris i desenvolupadors d'APIs una forma més senzilla de poder descobrir noves APIs i poder donar a conèixer les seves pròpies.

És per això que es pot considerar que milloraria la situació en aquest context de desenvolupament i descobriment d'APIs dins l'àmbit de l'enginyeria del software, però també és cert que no s'espera que tingui una repercussió important sobre la societat en general.

De totes maneres, es considera que la millora que pot aportar al subconjunt de la societat que representa els enginyers de software i en general, qualsevol persona interessada en la recerca i el desenvolupament d'APIs, fa mereixedor al projecte d'un 6 en la dimensió social.

5.3 Dimensió ambiental

Pel que fa a la dimensió ambiental, aquest projecte no contribuirà a augmentar la petjada ecològica, ja que es tracta simplement d'un servei online, però també és cert que no la reduirà.

No serà necessari, tampoc, la utilització de matèries primeres ni productes manufacturats per al desenvolupament del projecte, ni es preveu el desmantellament de cap recurs material un cop acabat.

És per això que es considera que aquest projecte puntua un 7 en l'àmbit de la sostenibilitat en l'àmbit ambiental.

6. Requisits del sistema

De cara a l'anàlisi de requisits s'ha decidit utilitzar la plantilla Volere per a especificació de requisits [16], tant per als requisits funcionals com pels no funcionals.

Els tipus indicats en les taules incloses en els dos apartats que hi ha a continuació són els següents:

- 9: Requisits funcionals
- 10: Requisits d'aparença i estil
    - 10a: Requisits d'aparença
    - 10b: Requisits d'estil
- 11: Requisits d'usabilitat i humanitat
    - 11a: Requisits de facilitat d'ús
    - 11d: Requisits de claredat
- 12: Requisits de rendiment
    - 12a: Requisits de velocitat i latència
    - 12d: Requisits de fiabilitat i disponibilitat
    - 12g: Requisits d'escalabilitat o extensibilitat
- 14: Requisits de mantenibilitat i suport
    - 14c: Requisits d'adaptabilitat
- 15: Requisits de seguretat
    - 15b: Requisits d'integritat
    - 15c: Requisits de privacitat

Aquestes taules també mostren el nivell de satisfacció (entre 1 i 5) del client en veure que el requisit en qüestió ha sigut implementat, el nivell d'insatisfacció (entre 1 i 5) que pot provocar en el client que el requisit en qüestió no sigui finalment implementat, les dependències que el requisit en qüestió crea sobre els altres, els conflictes que provoca amb altres requisits, i els materials de suport amb els quals compta.

6.1 Requisits funcionals

# Requisit:	1	**Tipus:**	9	**Casos d'ús relacionats:**	1
Descripció:	Permetre a qualsevol usuari veure un llistat de totes les APIs del sistema on, per a cada API, hi aparegui el seu nom, la seva descripció i les categories que té associades.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	El sistema mostra un llistat amb el nom, la descripció i les categories de totes les APIs que té emmagatzamades quan s'accedeix a la pàgina principal.				
Satisfacció pel client:	3	**Insatisfacció pel client:**	4		
Dependències:	2	**Conflictes:**	-		
Materials de suport:	Figura 6				

# Requisit:	2	**Tipus:**	9	**Casos d'ús relacionats:**	1, 2
Descripció:	Permetre a qualsevol usuari filtrar el llistat de totes les APIs per text o per categories associades a les APIs que s'hi mostren.				
Justificació:	Útil per a permetre als usuaris veure només aquelles APIs que realment els interessen.				
Criteri de satisfacció:	El sistema compta amb els elements necessaris per tal de poder filtrar de les dues formes: per text o seleccionant una categoria d'un llistat.				
Satisfacció pel client:	3	**Insatisfacció pel client:**	2		
Dependències:	-	**Conflictes:**	-		
Materials de suport:	Figura 6				

# Requisit:	3	**Tipus:**	9	**Casos d'ús relacionats:**	3
Descripció:	Permetre a qualsevol usuari veure la informació completa d'una API.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	El sistema mostra, per a cadascuna de les APIs tots els seus atributs: Nom, descripció, portal, protocol, format de request, format de response, endpoints, desenvolupadors, llibreries externes, pricing, disponibilitat, precisió i temps mitjà de resposta.				
Satisfacció pel client:	5	**Insatisfacció pel client:**	5		
Dependències:	4, 5	**Conflictes:**	-		
Materials de suport:	Figura 11				

# Requisit:	4	**Tipus:**	9	**Casos d'ús relacionats:**	3, 4
Descripció:	Permetre a l'usuari que ha introduït informació sobre una API al sistema, editar-la.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	Quan l'usuari que l'ha introduït visualitza la informació d'una API, el sistema li mostra la opció d'editar tots els camps d'aquesta API, a excepció del seu nom.				
Satisfacció pel client:	3	**Insatisfacció pel client:**	3		
Dependències:	-	**Conflictes:**	-		
Materials de suport:	Figura 12				

# Requisit:	5	Tipus:	9	Casos d'ús relacionats:	3, 6
Descripció:	Permetre a l'usuari que ha introduït informació sobre una API al sistema, esborrar-la per complert.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	Quan l'usuari que l'ha introduït visualitza la informació d'una API, el sistema li mostra l'opció d'esborrar tota la informació de l'API en qüestió.				
Satisfacció pel client:	2	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	Figura 11				

# Requisit:	6	Tipus:	9	Casos d'ús relacionats:	5
Descripció:	Permetre a un usuari registrat introduïr la informació d'una nova API al sistema.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	Quan l'usuari ha iniciat sessió, el sistema li mostra la opció per a introduïr una nova API. Quan la selecciona, pot omplir els camps que decideixi, a excepció del nom i una URL per a provar els requisits no funcionals, que sempre s'han d'incloure, i guardar la informació de l'API.				
Satisfacció pel client:	5	Insatisfacció pel client:	5		
Dependències:	1, 3	Conflictes:	-		
Materials de suport:	Figura 10				

# Requisit:	7	Tipus:	9	Casos d'ús relacionats:	7
Descripció:	Permetre a un usuari registrat visualitzar la seva informació o la d'un altre usuari registrat.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	Quan l'usuari ha iniciat sessió, el sistema li mostra l'opció de visualitzar el seu perfil. A més, si visita el perfil de qualsevol altre usuari, en podrà veure les seves dades públiques.				
Satisfacció pel client:	3	Insatisfacció pel client:	2		
Dependències:	8	Conflictes:	-		
Materials de suport:	Figura 9				

# Requisit:	8	Tipus:	9	Casos d'ús relacionats:	8
Descripció:	Permetre a un usuari registrat editar la seva informació.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	Quan l'usuari ha iniciat sessió, el sistema li mostra l'opció de visualitzar el seu perfil, i un cop allà, de modificar-ne el seu nom o la seva contrassenya.				
Satisfacció pel client:	2	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	Figura 9				

# Requisit:	9	Tipus:	9	Casos d'ús relacionats:	9
Descripció:	Permetre a un usuari registrar-se a la web i poder fer servir el conjunt de funcionalitats restringides als usuaris registrats.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	El sistema permet a qualsevol usuari que així ho vulgui, registrar-se a la pàgina triant un nom d'usuari, una contrassenya, i, opcionalment, introduïnt també el seu nom.				
Satisfacció pel client:	4	Insatisfacció pel client:	5		
Dependències:	10	Conflictes:	-		
Materials de suport:	Figura 7				

# Requisit:	10	Tipus:	9	Casos d'ús relacionats:	10
Descripció:	Permetre a un usuari registrat iniciar sessió i poder accedir així a totes les funcionalitats que ofereix el sistema.				
Justificació:	Necessari per a complir amb els objectius del projecte.				
Criteri de satisfacció:	El sistema permet a qualsevol usuari registrat iniciar sessió introduïnt la combinació de nom d'usuari i contrassenya triada en el moment del seu registre.				
Satisfacció pel client:	4	Insatisfacció pel client:	5		
Dependències:	6, 7	Conflictes:	-		
Materials de suport:	Figura 8				

# Requisit:	11	Tipus:	9	Casos d'ús relacionats:	11
Descripció:	Permetre a un usuari registrat i que tingui la sessió iniciada, tancar-la.				
Justificació:	Necessari perque l'usuari pugui esborrar les dades que es guarden a l'emmagatzematge local del navegador que estigui fent servir i mantenir així la seva privacitat.				
Criteri de satisfacció:	El sistema proporciona a qualsevol usuari registrat que hagi iniciat una sessió la opció de tancar-la.				
Satisfacció pel client:	2	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	12	Tipus:	9	Casos d'ús relacionats:	12
Descripció:	Permetre a un usuari registrat esborrar el seu compte.				
Justificació:	Necessari per a aquells usuaris que decideixin que no volen fer servir el sistema després d'haver-lo provat i no volen que les seves dades segueixin al sistema.				
Criteri de satisfacció:	El sistema permet a qualsevol usuari registrat donar-se de baixa del sistema des del seu perfil.				
Satisfacció pel client:	2	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

6.2 Requisits no funcionals

# Requisit:	13	**Tipus:**	10a	**Casos d'ús relacionats:**	-
Descripció:	El sistema ha de ser considerat atractiu pels seus usuaris potencials.				
Justificació:	Necessari per tal d'incentivar el seu ús per part d'usuaris potencials.				
Criteri de satisfacció:	Un 75%, com a mínim, dels usuaris que hi entren, es registrin o no, el troben atractiu.				
Satisfacció pel client:	3		**Insatisfacció pel client:**	2	
Dependències:	-		**Conflictes:**	-	
Materials de suport:	-				

# Requisit:	14	**Tipus:**	10b	**Casos d'ús relacionats:**	-
Descripció:	El sistema ha de tenir un disseny simple i minimalista.				
Justificació:	Aquest tipus de disseny és el que es porta entre la majoria de plataformes similars.				
Criteri de satisfacció:	Un 75%, com a mínim, dels usuaris del sistema consideren que el sistema és, efectivament, simple i minimalista.				
Satisfacció pel client:	3		**Insatisfacció pel client:**	2	
Dependències:	-		**Conflictes:**	-	
Materials de suport:	-				

# Requisit:	15	**Tipus:**	11a	**Casos d'ús relacionats:**	-
Descripció:	El sistema ha de ser fàcil de fer servir per a tots els usuaris que hi entrin, de forma que cap d'ells necessiti coneixements previs específics per tal de fer servir la pàgina (fora dels que es pugui necessitar per entendre tot allò relacionat amb les APIs).				
Justificació:	Que el sistema sigui fàcil d'utilitzar pot incentivar als usuaris perque continuïn fent-lo servir.				
Criteri de satisfacció:	Un 75%, com a mínim, dels usuaris que facin servir el sistema han de ser capaços de realitzar tots els casos d'ús de la web sense dificultat.				
Satisfacció pel client:	4		**Insatisfacció pel client:**	3	
Dependències:	-		**Conflictes:**	-	
Materials de suport:	-				

# Requisit:	16	**Tipus:**	11d	**Casos d'ús relacionats:**	-
Descripció:	El llenguatge usat al sistema ha de ser clar i concís.				
Justificació:	Necessari per evitar que cap usuari pateixi un malentès degut a part del llenguatge que pugui resultar ambigua.				
Criteri de satisfacció:	Un 75%, com a mínim, dels usuaris que facin servir el sistema afirmen que el llenguatge usat és clar i concís.				
Satisfacció pel client:	3		**Insatisfacció pel client:**	3	
Dependències:	-		**Conflictes:**	-	
Materials de suport:	-				

# Requisit:	17	Tipus:	12a	Casos d'ús relacionats:	-
Descripció:	Les crides fetes al sistema han de retornar una resposta en el mínim temps possible.				
Justificació:	Necessari per no provocar esperes innecessàries als usuaris.				
Criteri de satisfacció:	No hi ha cap crida a la API interna del sistema que trigui més de 2 segons en retornar una resposta.				
Satisfacció pel client:	4	Insatisfacció pel client:	3		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	18	Tipus:	12d	Casos d'ús relacionats:	-
Descripció:	El sistema ha de tenir una alta disponibilitat.				
Justificació:	Necessari per tal que qualsevol usuari pugui accedir-hi sempre que vulgui.				
Criteri de satisfacció:	El sistema té un mínim d'un 95% de temps en línia.				
Satisfacció pel client:	2	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	19	Tipus:	12g	Casos d'ús relacionats:	-
Descripció:	El sistema ha de ser fàcilment extensible.				
Justificació:	Necessari per si es decideix afegir-hi funcionalitats noves un cop la primera versió estigui acabada.				
Criteri de satisfacció:	En l'avaluació realitzada després del procés d'implementació es considera que l'aplicació és extensible.				
Satisfacció pel client:	1	Insatisfacció pel client:	1		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	20	Tipus:	14c	Casos d'ús relacionats:	-
Descripció:	El disseny del sistema ha d'adaptar-se a qualsevol dispositiu des del que s'hi accedeixi.				
Justificació:	Necessari per permetre als usuaris que accedeixin a la web des del dispositiu que prefereixin.				
Criteri de satisfacció:	Un 75%, com a mínim, dels usuaris de la web afirmaran que poden utilitzar el sistema indistintament des de dispositius de sobretaula i mòbils.				
Satisfacció pel client:	3	Insatisfacció pel client:	2		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	21	Tipus:	15b	Casos d'ús relacionats:	-
Descripció:	El sistema ha de comptar amb sistemes de control d'entrada de dades.				
Justificació:	Necessari per a evitar problemes al sistema deguts a qualsevol dada introduida per un usuari.				
Criteri de satisfacció:	Tots els formularis de la web tenen sistemes de control que comproven la integritat de les dades introduïdes.				
Satisfacció pel client:	1	Insatisfacció pel client:	1		
Dependències:	-	Conflictes:	-		
Materials de suport:	-				

# Requisit:	22	Tipus:	15c	Casos d'ús relacionats:	-
Descripció:	El sistema ha de protegir la privacitat de les dades personals del usuaris registrats.				
Justificació:	Necessari per tal de mantenir les dades dels usuaris protegides i augmentar així la seva confiança en el sistema.				
Criteri de satisfacció:	El sistema fa servir autenticació per accedir a les dades dels usuaris, i dins la base de dades, les contrassenyes es guarden encriptades.				
Satisfacció pel client:	3	**Insatisfacció pel client:**	2		
Dependències:	-	**Conflictes:**	-		
Materials de suport:	-				

7. Especificació del sistema

Aquest apartat és el primer a realitzar després de la planificació del projecte, on es decideixen les funcionalitats que s'hi implementen. En el cas d'aquest projecte, s'han decidit segons les funcionalitats implementades pels cinc repositoris escollits en l'estudi de l'estat de l'art.

Concretament, s'ha decidit implementar aquelles funcionalitats que apareixien en un mínim de dos repositoris, amb l'excepció de la funcionalitat de mostrar notícies, que s'ha decidit no implementar, ja que no és imprescindible per al compliment dels objectius del projecte, i la funcionalitat de mostrar totes les APIs del repositori, que tot i no estar implementada en un mínim de dos repositoris, s'ha considerat que era prou important per a implementar-la igualment.

Així, els casos d'ús [17] que s'han definit per al sistema són els següents:

- Llistar APIs
- Filtrar llistat d'APIs
- Visualitzar informació d'una API
- Editar informació d'una API
- Afegir una nova API
- Esborrar informació d'una API
- Visualitzar informació d'un usuari
- Editar informació d'un usuari
- Registrar-se
- Iniciar sessió
- Tancar sessió
- Esborrar compte d'usuari

A continuació es mostren un diagrama on es poden veure els casos d'ús segons l'actor que pot realitzar-los, i una explicació de cadascun d'aquests casos d'ús.

7.1 Diagrama de casos d'ús

Figura 3: Diagrama de casos d'ús

7.2 Descripció dels casos d'ús

7.2.1 Llistar APIs

Actor: Usuari registrat / usuari no registrat

Precondicions: -

Disparador: L'usuari vol visualitzar un llistat de totes les APIs que hi ha al sistema.

Escenari principal d'èxit:

1. L'usuari accedeix a la pàgina principal de la web.
2. El sistema mostra el llistat de totes les APIs que té emmagatzemades.

7.2.2 Filtrar llistat d'APIs

Actor: Usuari registrat / usuari no registrat

Precondicions: -

Disparador: L'usuari vol filtrar el llistat d'APIs que el sistema li retorna.

Escenari principal d'èxit:

1. L'usuari accedeix a la pàgina principal de la web.
2. El sistema mostra el llistat de totes les APIs que té emmagatzemades.
3. L'usuari realitza, com a mínim, una de les accions següents:
    a. L'usuari escriu el text que vol cercar en els resultats.
    b. L'usuari selecciona una categoria de les disponibles.
4. El sistema mostra el llistat de totes les APIs que compleixen els criteris seleccionats.

7.2.3 Visualitzar informació d'una API

Actor: Usuari registrat / usuari no registrat

Precondicions: -

Disparador: L'usuari vol consultar tota la informació d'una API.

Escenari principal d'èxit:

1. L'usuari realitza una de les accions següents:
    a. L'usuari selecciona una API del llistat de la pàgina principal.
    b. L'usuari navega a l'adreça de la API que vulgui, si la coneix.
2. El sistema mostra tota la informació que té sobre aquella API.

7.2.4 Editar informació d'una API

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.
2. L'API que es vol modificar ha hagut de ser introduïda al sistema per l'usuari.

Disparador: L'usuari vol editar tota la informació d'una API.

Escenari principal d'èxit:

1. L'usuari realitza una de les accions següents:
    a. L'usuari selecciona una API del llistat de la pàgina principal.

    b. L'usuari navega a l'adreça de la API que vulgui, si la coneix.
2. El sistema mostra tota la informació que té sobre aquella API.
3. L'usuari prem el botó "Edit".
4. L'usuari modifica els camps que desitgi.
5. L'usuari prem el botó "Save".
6. El sistema guarda els canvis realitzats a la base de dades.
7. El sistema redirigeix a l'usuari a la pàgina amb la informació de l'API.

7.2.5 Afegir una nova API

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.

Disparador: L'usuari vol afegir la informació d'una nova API al sistema.

Escenari principal d'èxit:

1. L'usuari navega fins a la pestanya corresponent a Afegir una nova API.
2. L'usuari omple els camps del formulari que apareix amb la informació de l'API.
3. L'usuari prem el botó "Guardar".
4. El sistema guarda la informació a la base de dades.
5. El sistema actua segons si es produeix algun error o no:
    a. Si es produeix algun error, el sistema el notifica a l'usuari.
    b. Si no hi ha cap error, el sistema redirigeix a l'usuari a la pàgina corresponent a la informació de l'API afegida.

7.2.6 Esborrar informació d'una API

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.
2. L'API que es vol esborrar ha hagut de ser introduïda al sistema per l'usuari.

Disparador: L'usuari vol esborrar tota la informació d'una API.

Escenari principal d'èxit:

1. L'usuari realitza una de les accions següents:
    a. L'usuari selecciona una API del llistat de la pàgina principal.
    b. L'usuari navega a l'adreça de la API que vulgui, si la coneix.

2. El sistema mostra tota la informació que té sobre aquella API.
3. L'usuari fa click al botó "Delete".
4. El sistema mostra un popup demanant a l'usuari que confirmi que vol esborrar la API.
5. L'usuari fa click al botó "Delete" del popup.
6. El sistema esborra l'API de la base de dades.
7. El sistema redirigeix a l'usuari a la pàgina principal.

7.2.7 Visualitzar informació d'un usuari

Actor: Usuari registrat / usuari no registrat

Precondicions: -

Disparador: L'usuari vol visualitzar la informació d'un usuari (que pot o no ser ell mateix).

Escenari principal d'èxit:

1. L'usuari accedeix directament al perfil de l'usuari del qual vol visualitzar la informació d'una de les formes següents:
    a. Mitjançant l'adreça del perfil.
    b. Fent click a la pestanya "Profile" si vol accedir al seu perfil.
2. El sistema mostra la informació sobre l'usuari en qüestió.

7.2.8 Editar informació d'un usuari

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.
2. La informació que es vol modificar ha de ser la de l'usuari que té la sessió iniciada.

Disparador: L'usuari vol editar la seva informació personal.

Escenari principal d'èxit:

1. L'usuari accedeix al seu perfil fent click sobre la pestanya "Profile".
2. El sistema mostra la informació personal de l'usuari.
3. L'usuari modifica el seu nom o la seva contrasenya.
4. L'usuari prem el botó "Save".
5. El sistema guarda els canvis a la base de dades.
6. El sistema actua segons si es produeix algun error o no:
    a. Si es produeix algun error, el sistema el notifica a l'usuari.

b. Si no hi ha cap error, el sistema informa a l'usuari que els canvis han estat realitzats correctament.

7.2.9 Registrar-se

Actor: Usuari no registrat.

Precondicions:

1. L'usuari no pot estar registrat al sistema.

Escenari principal d'èxit:

1. L'usuari accedeix a la pestanya "Register".
2. L'usuari introdueix al formulari que hi apareix el nom d'usuari i la contrasenya que desitja fer servir per a registrar-se.
3. Si ho vol, l'usuari introdueix al formulari el seu nom real.
4. L'usuari prem el botó "Save".
5. El sistema guarda els canvis a la base de dades.
6. El sistema actua segons si es produeix algun error o no:
    a. Si es produeix algun error, el sistema el notifica a l'usuari.
    b. Si no hi ha cap error, el sistema redirigeix a l'usuari per tal que realitzi el cas d'ús "Iniciar Sessió".

7.2.10 Iniciar Sessió

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'estar registrat.
2. L'usuari no ha d'haver iniciat sessió.

Escenari principal d'èxit:

1. L'usuari accedeix a la pestanya "Login".
2. L'usuari introdueix al formulari el seu nom d'usuari i la seva contrasenya.
3. L'usuari prem el botó "Login".
4. El sistema comprova les credencials de l'usuari.
5. El sistema actua segons si es produeix algun error o no:
    a. Si es produeix algun error, el sistema el notifica a l'usuari.
    b. Si no hi ha cap error, el sistema redirigeix a l'usuari a la pàgina principal de la web.

7.2.11 Tancar sessió

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.

Escenari principal d'èxit:

1. L'usuari fa click a la pestanya "Logout".
2. El sistema tanca la sessió de l'usuari.
3. El sistema actua segons si es produeix algun error o no:
    a. Si es produeix algun error, el sistema el notifica a l'usuari.
    b. Si no hi ha cap error, el sistema redirigeix a l'usuari a la pàgina principal de la web.

7.2.12 Esborrar un usuari

Actor: Usuari registrat

Precondicions:

1. L'usuari ha d'haver iniciat sessió.
2. Un usuari només pot esborrar el seu propi compte.

Disparador: L'usuari vol donar-se de baixa del sistema.

Escenari principal d'èxit:

1. L'usuari accedeix directament al seu perfil d'una de les formes següents:
    a. Mitjançant l'URL de l'usuari.
    b. Fent click a la pestanya "Profile".
2. El sistema mostra a l'usuari la seva informació.
3. L'usuari fa click al botó "Delete".
4. El sistema mostra un popup demanant a l'usuari que confirmi que vol esborrar les seves dades.
5. L'usuari fa click al botó "Delete" del popup.
6. El sistema esborra l'usuari de la base de dades.
7. El sistema tanca la sessió de l'usuari.
8. El sistema redirigeix a l'usuari a la pàgina principal.

7.3 Model conceptual

En aquest projecte hi ha diverses entitats: les APIs, els usuaris, les categories, els desenvolupadors i els endpoints.

Les APIs s'identifiquen pel seu slug, que és la representació optimitzada per qüestions de brevetat i llegibilitat del seu nom. Concretament, aquest slug té totes les seves lletres en minúscula, els caràcters especials com vocals amb accents canviats per les vocals sense accents i els espais canviats per guions. A més, també es guarda de les APIs la seva descripció, el seu portal (URL base), l'URL que es fa servir per a calcular els valors dels seus requisits no funcionals, el protocol que fa servir, el format que accepta, tant per a request com per a response i el preu que tingui fer-la servir (en cas que no sigui gratuit).

A més, cada API està relacionada amb els seus desenvolupadors, dels quals es guarda el nom i la seva pàgina web; amb els seus endpoints, que compten amb un nom i una descripció cadascun; amb les llibreries externes en les quals es fa servir l'API, de les quals s'indica el nom i l'URL; amb les categories que té associades, que estan representades per un conjunt de strings i amb l'usuari que les introdueix al sistema.

Pel que fa als usuaris, el seu model és molt més simple: només es guarda el seu nom d'usuari, pel qual s'identifiquen, la seva contrasenya, i opcionalment, el seu nom real.

Les categories són simplement paraules clau que s'associen a les APIs per tal de poder classificar-les o agrupar-les segons similituds.

A continuació es pot veure la representació UML del model del projecte:

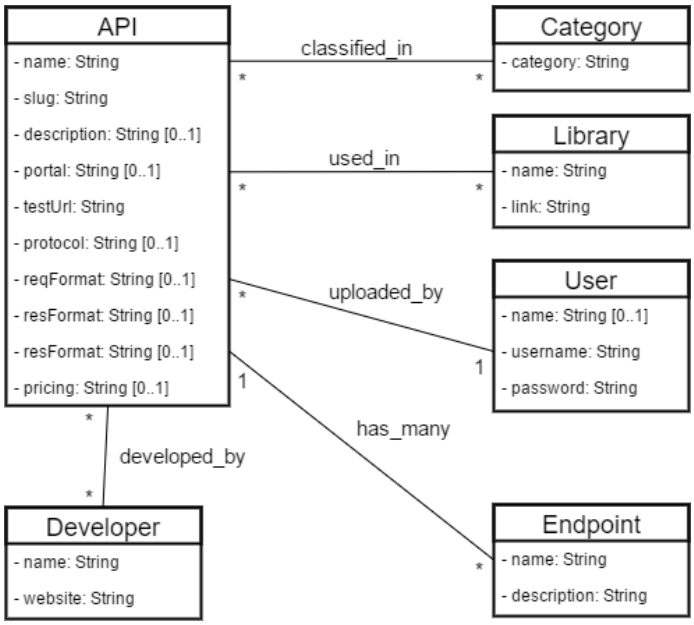


Figura 4: Model conceptual

8. Disseny del sistema

Un cop identificats els requisits del sistema, tant funcionals com no funcionals, i feta l'especificació del sistema, se'n detalla el disseny. Concretament, se'n mostra el disseny extern, durant el qual s'han definit les vistes del sistema, així com la navegabilitat entre elles, i el disseny intern, que compta amb decisions sobre l'arquitectura a utilitzar i el model de les entitats que es guarden al sistema.

8.1 Disseny extern

El disseny extern del sistema és el corresponent al mapa navegacional que es mostra a continuació.

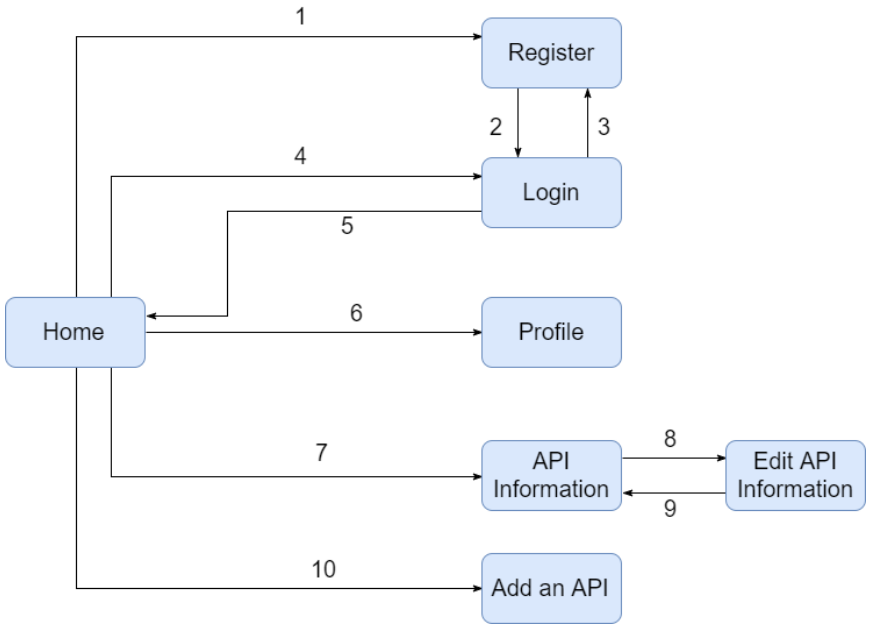


Figura 5: Mapa navegacional

Les accions que permeten navegar entre les vistes són les següents:

1. Fer click a l'opció "Register" que hi ha al header. Només disponible quan l'usuari no ha iniciat sessió.
2. Fer click a l'opció "Already have an account? Login here" o bé de forma automàtica un cop s'ha fet click al botó "Register" i el procés de registre s'ha completat correctament.
3. Fer click a l'opció "Not registered? Create your account here".
4. Fer click a l'opció "Login" que hi ha al header. Només disponible quan l'usuari no ha iniciat sessió.
5. De forma automàtica un cop s'ha fet click al botó "Login" i l'usuari s'ha autenticat correctament.

6. Fer click a l'opció "Profile" que hi ha al header. Només disponible quan l'usuari ha iniciat sessió.
7. Fer click al nom de qualsevol de les APIs que surten al llistat.
8. Fer click al botó "Edit".
9. Fer click al botó "Save".
10. Fer click a l'opció "Add an API" que hi ha al header. Només disponible quan l'usuari ha iniciat sessió.

A part, es pot tornar a la pàgina inicial des de qualsevol de la resta de pàgines mitjançant el header, que és visible des de totes elles.

A continuació s'inclouen captures de totes les vistes:

Home

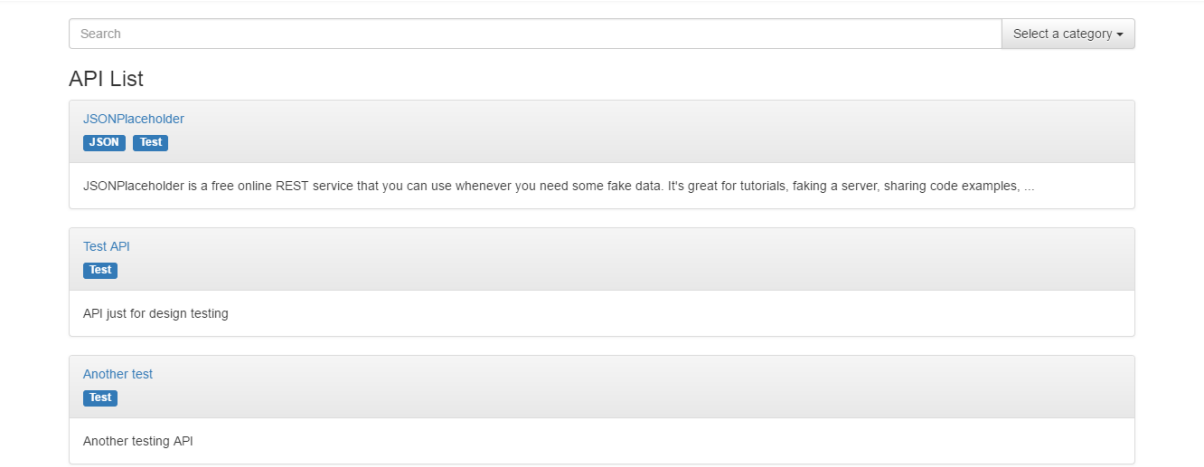


Figura 6: Pantalla 'Home'

Aquesta pantalla és la que serveix de pàgina principal de la web, i on es pot veure el llistat de les APIs que hi ha al sistema. Per a cada API es mostra el seu nom, les seves categories i la seva descripció. A més, la barra de cerca superior permet filtrar els resultats per text o bé per categoria, si se'n selecciona una al desplegable de la part dreta.

Register

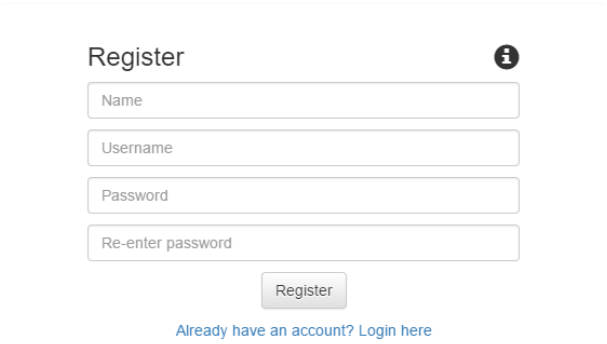


Figura 7: Pantalla 'Register'

Aquesta és la pantalla que qualsevol persona que entri a la web pot fer servir per a registrar-s'hi. Per a fer-ho ha d'introduir el seu nom (opcionalment), un nom d'usuari i una contrasenya. A part del formulari de registre, aquesta pantalla també té un enllaç a la pantalla d'inici de sessió, per si l'usuari ja està registrat, i una icona d'informació, on es diu a l'usuari que les dades que introdueixi en el registre seran usades només per a permetre l'autenticació i l'accés a totes les funcionalitats reservades per a usuaris registrats.

Login

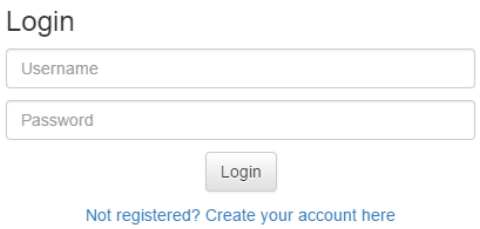


Figura 8: Pantalla 'Login'

En aquesta pantalla, qualsevol usuari que s'hagi registrat prèviament pot iniciar sessió omplint el formulari.

Profile

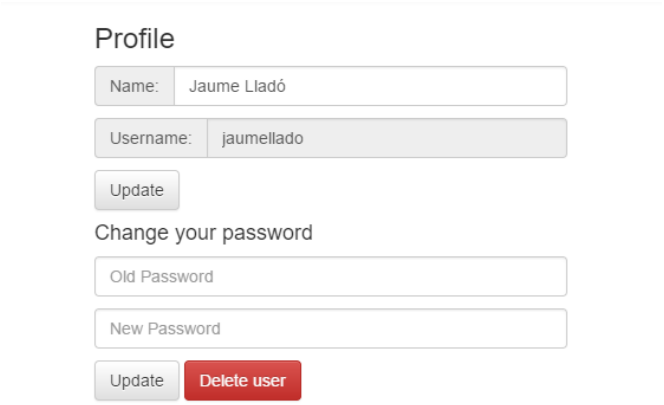


Figura 9: Pantalla 'Profile'

A la pantalla de perfil, un usuari pot modificar el seu nom i la seva contrassenya. A més, si ho decideix, també es pot donar de baixa.

New API

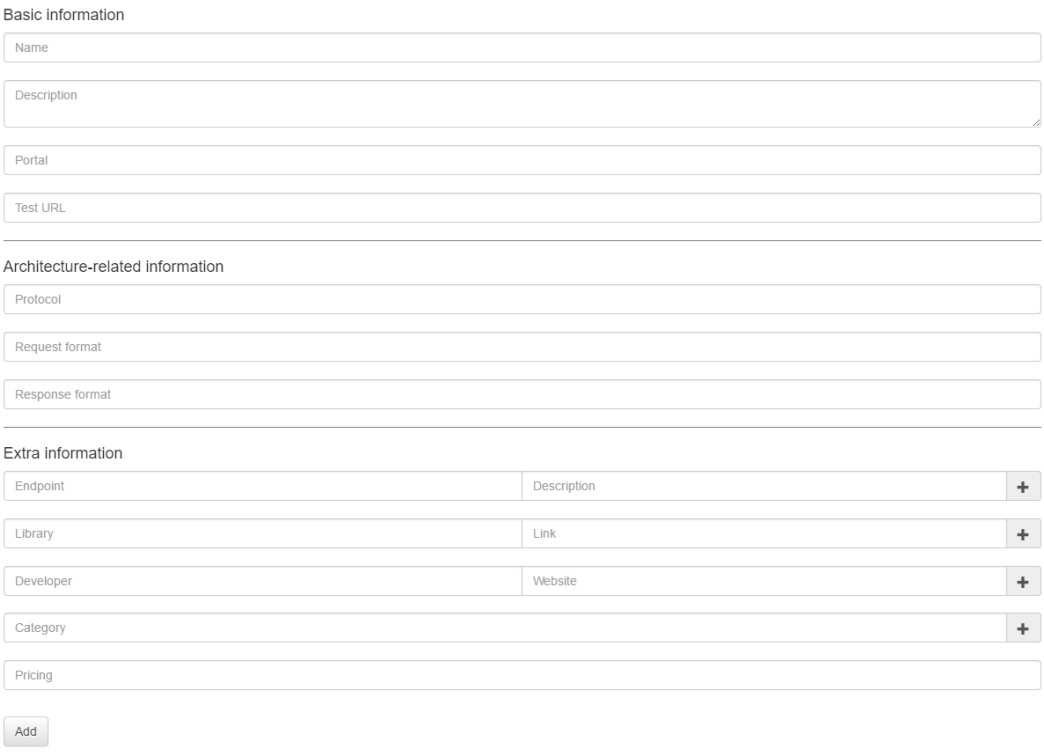


Figura 10: Pantalla 'New API'

Aquesta és la pantalla que un usuari pot fer servir per a afegir informació sobre una nova API al sistema. De tots els camps, els mínims requerits per tal de poder afegir l'API són el nom i l'URL de test.

API Information

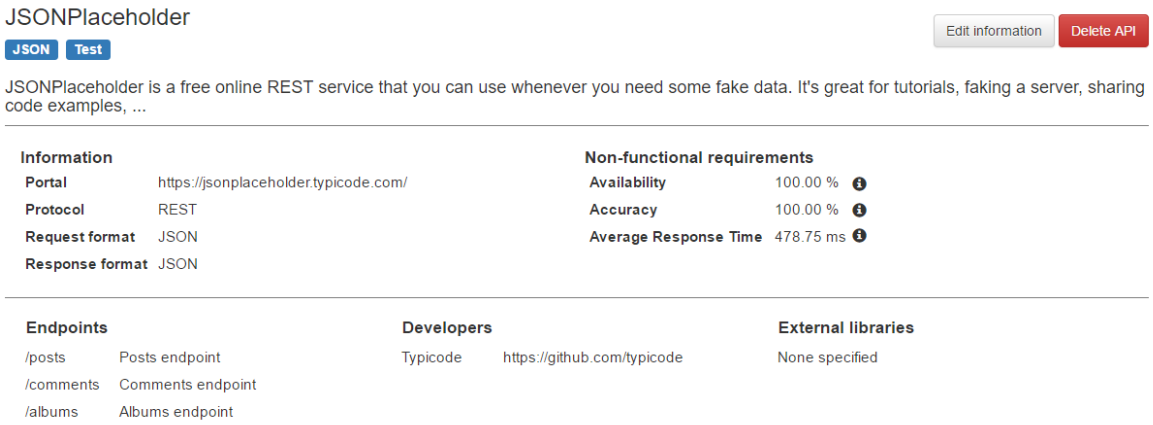


Figura 11: Pantalla 'API Information'

Aquesta és la pantalla corresponent a la informació d'una API. Si l'usuari que la visita és el mateix que ha introduït l'API en qüestió al sistema, s'habiliten els dos botons del final, que permeten editar la informació de l'API o esborrar-la, respectivament.

API Information Edit

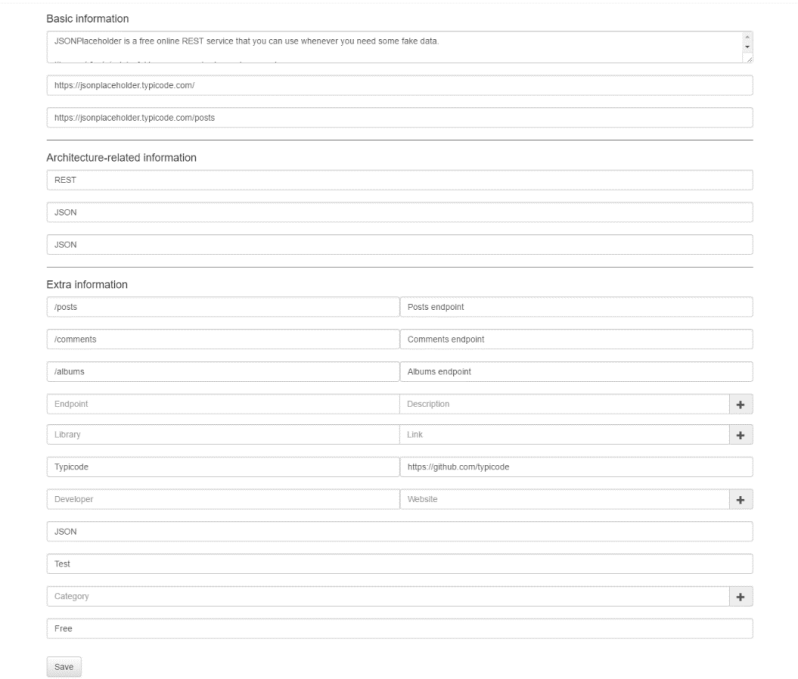


Figura 12: Pantalla 'Edit API Information'

En aquesta pantalla es pot editar tota la informació d'una API de forma similar a la pantalla d'afegir una nova API, i a excepció del seu nom. Un cop modificats els camps que siguin, només cal prémer el botó Save per tal de guardar els canvis i tornar automàticament a la pàgina d'informació de l'API.

8.2 Disseny intern

Per a aquesta aplicació s'ha decidit fer servir una arquitectura client/servidor per a pàgines web basada en el patró MVC (Model-Vista-Controlador) [18].

Els avantatges d'usar el patró MVC són la modularitat que ofereix, ja que separa les vistes de la lògica de negoci i les dades, de forma que el codi queda generalment net i fàcil de mantenir, l'extensibilitat que permet aconseguir, ja que es fa servir una única vista principal a la qual s'injecten les que faci falta segons el lloc de la web al qual es navegui, i la reusabilitat que ofereix, ja que es pot fer servir el mateix model de dades en diverses vistes sense problemes.

En aquesta arquitectura, la comunicació entre els tres elements que la componen es realitza amb els controladors com a intermediaris de la comunicació entre vistes i models. Concretament, la vista comunica al controlador que l'usuari ha realitzat alguna acció sobre la interfície de l'aplicació i aquest fa una petició al model per tal que s'actualitzi i els canvis que s'han realitzat a la vista hi quedin reflectits. Un cop guardats els canvis, el model notifica al controlador, i aquest és l'encarregat de comunicar a la vista que els canvis ja s'han guardat, i que per tant, pot actualitzar-se per tal de mostrar l'estat actual de les dades. Tot això es pot veure esquematitzat en la figura següent.

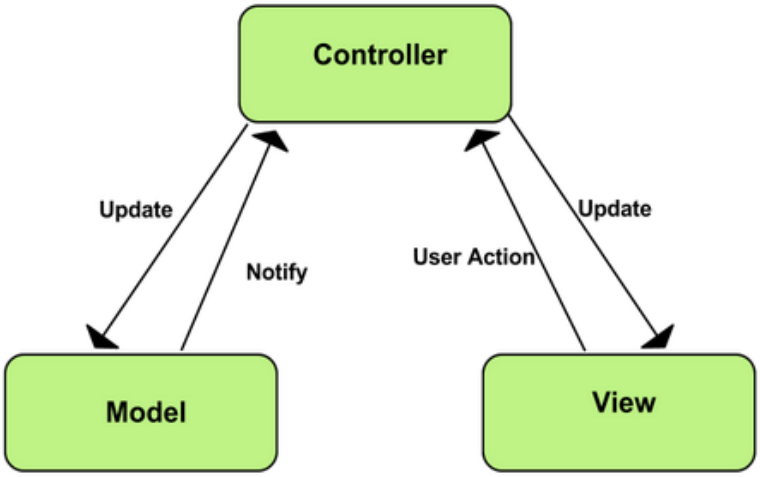


Figura 13: Patró MVC

En el cas del projecte, l'esquema amb tots els components seguint el patró MVC és el següent:

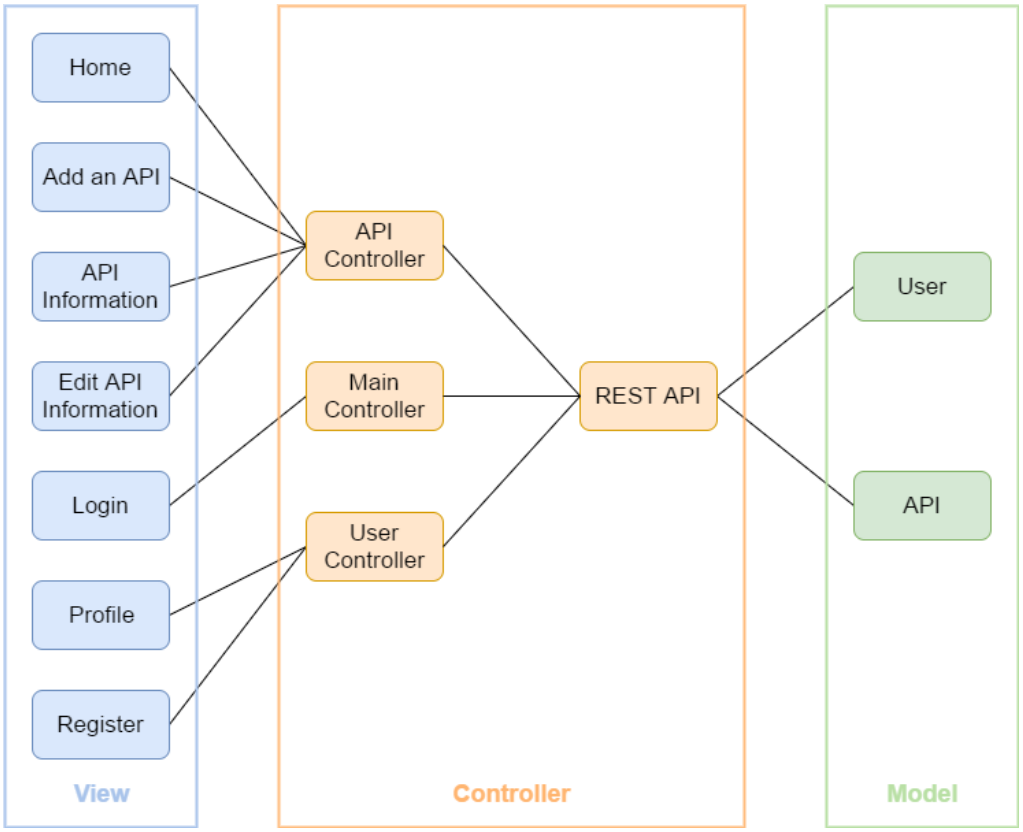


Figura 14: Aplicació de MVC al projecte

8.2.1 Vista

La vista d'aquesta aplicació és la corresponent a una Single Page Application (SPA). Gràcies a la tecnologia AJAX (Asynchronous JavaScript and XML), les aplicacions web actuals es poden desenvolupar de forma que es puguin enviar peticions addicionals al servidor un cop finalitzada la primera, de forma que el contingut d'una única pàgina es pot anar refrescant dinàmicament segons les peticions realitzades, i sense necessitat de refrescar la pàgina web sencera, com passava en les webs desenvolupades de forma tradicional.

8.2.2 Controlador

Aquesta part de l'aplicació és la que s'encarrega d'implementar la lògica de negoci. En el cas d'aquesta aplicació, aquest controlador serà una API REST.

Una API REST (REpresentational State Transfer) és una API implementada amb una arquitectura client/servidor sense estat que treballa amb protocol HTTP. Les característiques d'una API REST són les següents:

- Tots els recursos del sistema tenen el seu propi URI (Uniform Resource Identifier), que permet identificar-los i distingir-los de la resta d'elements.

- Hi ha una separació entre client i servidor que ajuda a millorar la portabilitat d'aquestes API.
- No té estat, per tant, cada petició HTTP ha de contenir tota la informació necessària per ser entesa sense necessitar cap tipus de context guardat a la part del servidor.
- Treballa sobre HTTP fent servir els quatre mètodes que proporciona HTTP (GET, POST, PUT i DELETE) per a manipular els recursos que conté.

La API REST implementada per al projecte es detalla de forma completa a l'annex corresponent.

8.2.3 Model

Per a aquest projecte, totes les instàncies de qualsevol dels dos models seran guardats com a objectes en format JSON (JavaScript Object Notation) [19], un llenguatge altament utilitzat per a l'intercanvi de dades en aplicacions web.

De les sis entitats definides inicialment n'han quedat dues, els Users i les APIs. La informació corresponent a Categories, Endpoints, Libraries i Developers es guarda com a conjunts dins els objectes de les APIs.

A continuació es pot veure com són els models de les APIs i els usuaris de l'aplicació:

API.json

```
{
  name: { type: String, required: true},
  slug: { type: String, unique: true },
  description: String,
  tags: Array,
  portal: String,
  testUrl: String,
  endpoints: Array,
  protocol: String,
  reqFormat: String,
  resFormat: String,
  libs: Array,
  devs: Array,
  pricing: String,
  totalReq: { type: Number, default: 0 },
  totalRes: { type: Number, default: 0 },
  correctRes: { type: Number, default: 0 },
  totalResTime: { type: Number, default: 0 },
  user: {type: String, required: true}
}
```

En l'atribut slug, el camp 'unique : true' indica que no hi podrà haver dos documents d'API que tinguin el mateix slug. Per tant, qualsevol API podrà ser identificada pel seu slug.

User.json

```
{
  name: String,
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true, select: false }
}
```

En els atributs username i password, el camp 'required:true ' implica que qualsevol usuari que es vulgui registrar a la web està obligat a triar un nom d'usuari (que no es podrà repetir pel camp 'unique: true') i una contrassenya. El camp 'select: false' implica que no es retornarà mai la contrassenya d'un usuari en obtenir el document que el representa, a no ser que es demani explícitament, com cal fer en alguna de les crides que hi ha a la API.

8.2.4 Exemple de diagrama de seqüència

Amb aquesta arquitectura, el cas d'ús de visualitzar la informació d'una API es representaria de la següent forma:

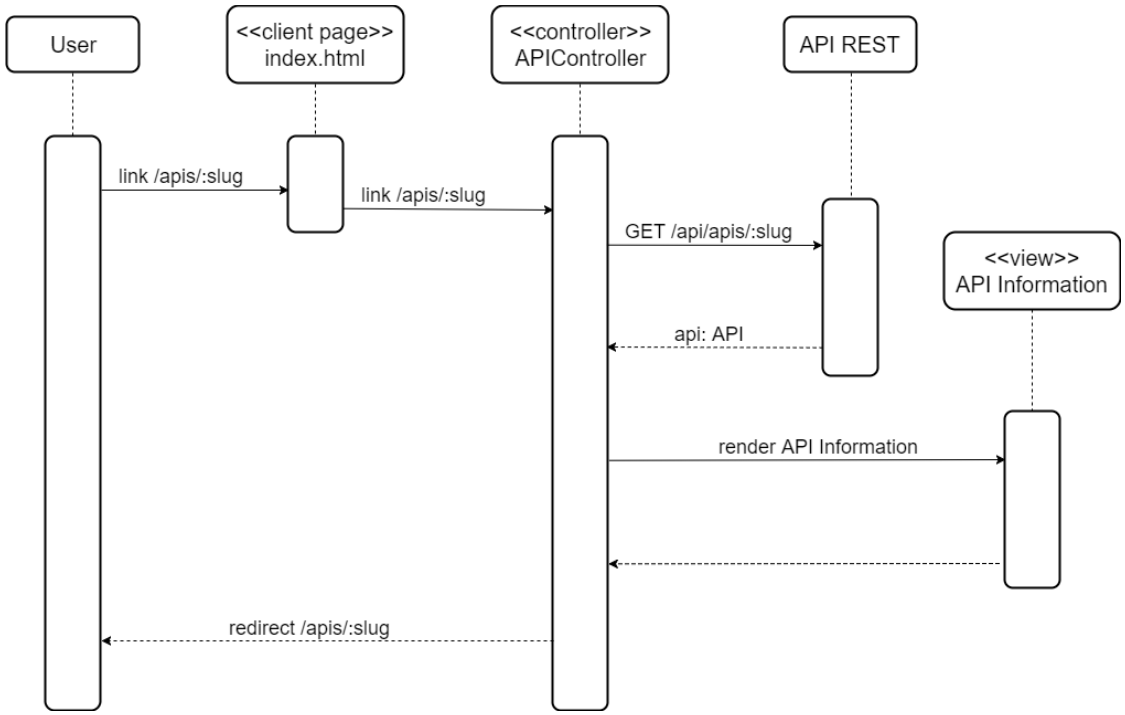


Figura 15: Exemple de diagrama de seqüència

Com es pot veure, quan l'usuari fa click a una de les APIs de la llista que es mostra a la pàgina principal, o bé quan introdueix l'URL corresponent a aquella API, la vista fa la petició al controlador, que demana la informació de l'API a l'API REST del sistema. Un cop la té, el mateix controlador renderitza la vista en qüestió i hi redirigeix l'usuari.

9. Implementació

Per al desenvolupament de la web s'ha decidit fer servir el que es coneix com a MEAN Stack, que consisteix en un grup de tecnologies (MongoDB, Express.js, AngularJS i Node.js) que es caracteritzen per treballar molt bé juntes. Això ve donat, en gran part, pel fet que amb un únic llenguatge de programació, JavaScript, es pot escriure el codi necessari per a qualsevol dels quatre elements.

9.1 Tecnologies usades

9.1.1 HTML

HyperText Markup Language (HTML) [20] és el llenguatge de marcat utilitzat per a la creació de pàgines web. La seva funcionalitat és descriure l'estructura de la pàgina web, sense tenir en compte aparença ni funcionalitat. Per a fer-ho, HTML utilitza etiquetes (tags), que permeten definir elements com títols, paràgrafs, seccions o imatges, entre d'altres, dins una pàgina web. Actualment, HTML és considerada una eina estàndard per a l'elaboració de pàgines web segons el World Wide Web Consortium (W3C).

9.1.2 CSS

Per tal de modificar l'aparença que HTML no té en compte es fa servir el que es coneix com a Cascading Style Sheets (CSS) [21] un altre element bàsic en el desenvolupament de pàgines web. CSS té com a finalitat descriure com han de mostrar-se els elements que hi ha estructurats en un fitxer HTML. Utilitzar-lo comporta diversos avantatges, ja que permet fer les pàgines web molt més atractives del que es pot aconseguir sense utilitzar-lo, i a més, permet estalviar grans quantitats de temps gràcies a la seva reusabilitat, ja que amb un únic document en format CSS es pot donar estil a diverses pàgines HTML.

9.1.3 JavaScript

JavaScript [22] és un llenguatge de programació orientada a objectes que s'encarrega d'afegir a HTML les funcionalitats que hi manquen inicialment, i que normalment s'utilitza en la part del client en una pàgina web d'estil client/servidor. Usar JavaScipt permet manipular els elements de la pàgina web o comunicar-se amb el servidor per tal de generar el contingut dinàmic de la pàgina web, entre d'altres.

En el cas del sistema que s'implementa en aquest projecte, JavaScript és de vital importància, ja que el framework de desenvolupament triat, MEAN, utilitza JavaScript en totes les tecnologies que el componen.

9.1.4 JSON

JavaScript Object Notation (JSON) [19] és un format de text lleuger que s'utilitza per a l'intercanvi de dades. Actualment, és un llenguatge altament utilitzat, ja que per la seva definició, és molt senzill d'analitzar i interpretar, tant per humans com per màquines, fet que el converteix en un llenguatge ideal per a l'intercanvi de dades. En aquesta aplicació, JSON és el format utilitzat per transmetre dades de l'API interna als controladors de cadascuna de les vistes i per a guardar la informació de les APIs i usuaris del repositori, en forma de documents d'una base de dades NoSQL.

9.1.5 MongoDB

MongoDB [23] és una base de dades NoSQL open-source, que emmagatzema la informació en forma de documents JSON. Que sigui NoSQL fa que no tingui un esquema definit, és a dir, que l'estructura dels documents que emmagatzema no hagi de ser la mateixa per tots. En el cas de MEAN Stack, però, es fa servir Mongoose, un driver per a la connexió entre Node.js i MongoDB que fa d'ODM (Object Data Mapper) i que, per tant, necessita que hi hagi un esquema. En conseqüència, l'únic aspecte aprofitable de no tenir esquema és que, encara que hagin de ser els mateixos per a tots els documents d'una col·lecció, els atributs d'un document poden variar pel que fa a mida. Tot i això, se segueix aprofitant la resta d'avantatges que aporten MongoDB i Mongoose com és el mapeig directe d'objectes JSON entre la base de dades i la API, ja que ambdues comparteixen el format, l'ús d'índexs i agregació en temps real per a agilitzar consultes, la facilitat d'ús que els caracteritza, i el fet que Mongoose utilitzi JavaScript, en comú amb la resta d'elements de MEAN.

9.1.6 Express.js

Express.js [24] és el framework estàndard que es fa servir en l'àmbit de servidor en aplicacions que utilitzen Node.js per tal de redirigir les crides que rep l'API de l'aplicació cap als endpoints corresponents. A més, Express és també l'encarregat de definir l'API en si, així com els middleware necessaris per a l'autenticació d'usuaris, en el cas de l'API d'aquest projecte.

9.1.7 AngularJS

AngularJS [25] és el framework que s'utilitza a la part del client de les aplicacions web desenvolupades utilitzant MEAN Stack. És un software open-source mantingut per Google que ha esdevingut molt popular en els últims anys en la creació de pàgines webs d'estil SPA, ja que permet extendre les vistes HTML afegint-hi funcionalitats gràcies a la programació en JavaScript.

De les característiques que ofereix AngularJS, cal destacar-ne el que es coneix com a "data-binding", que consisteix en la sincronització automàtica entre les dades de les vistes de l'aplicació i el model corresponent, que permet crear les pàgines SPA amb contingut generat dinàmicament sense haver de refrescar pàgines web al navegador.

En l'aplicació, AngularJS també es fa servir per a la definició de l'aplicació en si, per a injectar els mòduls necessaris per al funcionament del sistema, per a definir les rutes de l'aplicació, és a dir, per a dir quina vista s'ha de renderitzar segons la ruta de la web a la qual s'accedeixi i per a afegir un interceptador HTTP a les crides que es fan des del client al servidor que permet no haver d'afegir els elements d'autenticació necessaris per a la correcta resolució de les crides en cadascuna d'elles.

9.1.8 Node.js

Per últim, Node.js [26] és l'entorn d'execució sobre el qual es desenvolupa l'aplicació, i el que combina la resta d'elements de MEAN. Es caracteritza per ser asíncron i "event-driven", és a dir, els servidors implementats amb aquesta tecnologia no esperen mai una resposta després d'una crida, sinó que segueixen fent el que toqui fins que l'API (o on sigui que fan la crida) respongui amb un esdeveniment per comunicar que les dades que es demanaven ja estan disponibles, moment en el qual el servidor les recull. A més, cal destacar que en estar basat en el Javascript Engine V8 de Google Chrome, Node.js és molt ràpid pel que fa a execució de codi.

Una altra gran utilitat de Node.js és npm (Node Package Manager) que permet afegir llibreries externes als projectes realitzats amb Node.js de forma ràpida i senzilla, i que en aquest projecte en concret s'ha usat per afegir alguns dels components que es mencionen en el següent apartat.

9.2 Altres components

9.2.1 Heroku

Heroku [27] és la plataforma cloud que s'ha triat com a servidor per al desplegament de l'aplicació en línia. Ha estat escollit principalment per la seva fàcil integració amb el repositori de codi ja existent del projecte a Bitbucket i per l'alta disponibilitat que ofereix.

9.2.2 Atom

Atom és l'editor de text de l'empresa de control de versions de software Github. Ha sigut l'eina que s'ha fet servir durant tota l'etapa de desenvolupament de l'aplicació.

9.2.3 Bitbucket

Bitbucket [12] és el software de control de versions que ofereix l'empresa de software Atlassian i que en aquest projecte s'ha fet servir per tal de disposar d'un repositori on penjar el codi per tal de controlar-ne les versions i facilitar el seguiment del projecte per part dels directors.

9.2.4 Bootstrap

Bootstrap [28] és un framework open-source que s'utilitza per a la part de front-end [29], principalment per a agilitzar les tasques de disseny de les vistes HTML d'una pàgina web. Això s'aconsegueix gràcies a les plantilles HTML i CSS que ofereix, i que es poden utilitzar de forma senzilla i gratuïta per tal de no haver de dissenyar un a un els elements que es volen mostrar en les interfícies de les aplicacions web.

9.2.5 jQuery

jQuery és una llibreria escrita en JavaScript que permet ampliar i alhora facilitar les funcionalitats que aquest llenguatge ofereix pel que fa a manipulació d'elements de documents HTML, animacions, AJAX o processament d'esdeveniments.

9.2.6 JWT

JSON Web Tokens (JWT) és una llibreria externa, afegida al projecte mitjançant npm, que defineix la forma de transmetre informació en format JSON de forma segura segons l'estàndard RFC 7519. Aquesta informació que es transmet pot ser verificada, ja que està digitalment signada, en el cas del projecte amb una clau privada.

9.2.7 Bcrypt-nodejs

Bcrypt-nodejs és una llibreria externa, afegida al projecte mitjançant npm, que permet encriptar informació. En aquest projecte s'ha utilitzat per a encriptar les contrasenyes dels usuaris registrats a l'aplicació.

9.2.8 Body-parser

Body-parser és una llibreria externa, afegida al projecte mitjançant npm, que permet facilitar la interpretació dels elements que es passen en el Body de les crides de tipus POST que es fan a l'API.

9.2.9 Mocha

Mocha [30] és un framework de test per JavaScript que permet fer proves tant sobre Node.js com sobre el navegador directament. Per al projecte s'ha utilitzat per a realitzar totes les proves un cop finalitzada la fase d'implementació.

9.2.10 Chai

Chai [31] és una llibreria d'assercions per proves de BDD [32] i TDD [33] sobre Node.js o navegador, que serveix principalment per a complementar els frameworks de proves. En el cas del projecte, s'ha utilitzat per a comprovar amb assercions els resultats de les proves fetes amb Mocha. Aquesta llibreria s'ha utilitzat junt amb Chai-HTTP per a poder fer les proves sobre la API REST del sistema.

9.3 Procés d'implementació

9.3.1 Plantejament

El primer que s'ha fet en aquesta fase ha sigut decidir la tecnologia que es faria servir per a crear la pàgina web, que ha acabat sent MEAN Stack. Un cop decidit això, s'ha preparat l'entorn de desenvolupament amb Node.js i l'editor de text Atom, s'ha creat el repositori corresponent a Bitbucket, l'eina triada per al control de versions, i s'ha començat a desenvolupar.

Un cop triada la tecnologia, s'ha adaptat l'esquema del sistema dissenyat a MEAN, resultant en el següent:

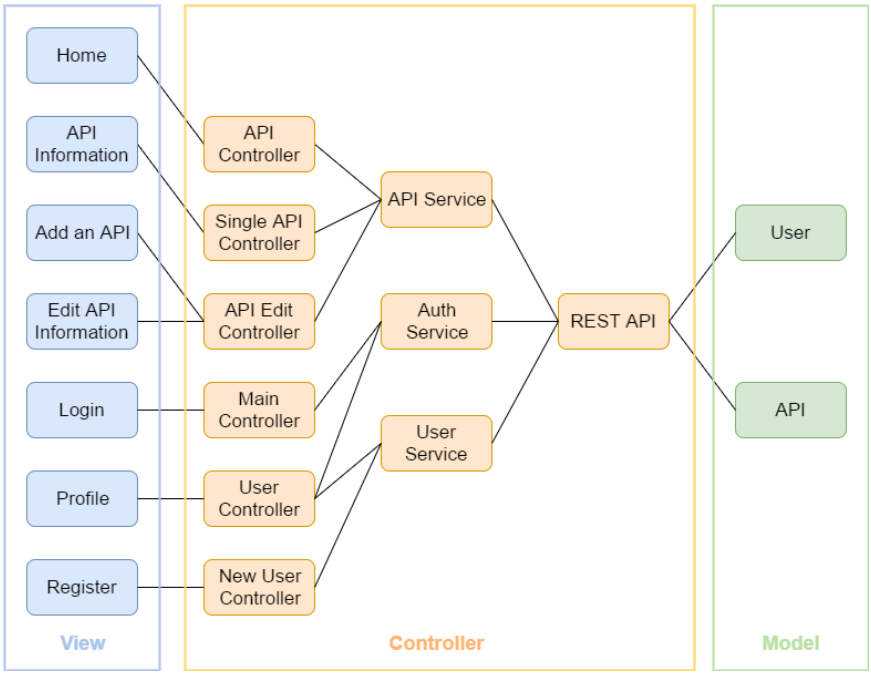


Figura 16: Patró MVC amb components desenvolupats

Concretament, es pot veure com els 3 controladors que hi havia inicialment (API Controller, User Controller i Main Controller) han esdevingut 7 controladors, de cara a augmentar la modularitat del sistema. Amb la mateixa idea s'han decidit afegir els serveis al sistema, de cara a aïllar els controladors associats a les vistes de l'API. API Service és l'encarregat d'executar les crides a l'API REST del projecte relacionades amb el model d'APIs, User Service fa el mateix per a aquelles crides relacionades amb el model d'User, i Auth Service és l'encarregat de gestionar tot el referent a l'autenticació d'usuaris.

Pel que fa a la base de dades, com que, tot i que MongoDB és NoSQL, Mongoose necessita tenir un esquema per poder fer el mapping d'objectes a documents JSON. Aquest esquema correspon exactament al mostrat en l'apartat de Model.

9.3.2 Desenvolupament

Per a la implementació del software del projecte s'ha acabat triant una adaptació de la metodologia àgil SCRUM. Ha hagut de ser una adaptació principalment pel fet que en lloc de comptar amb un equip de desenvolupament, aquest projecte ha estat desenvolupat íntegrament per una única persona, fet que també ha impedit la realització de daily scrums, pensats per a posar en comú la feina feta pels membres de l'equip i planejar les 24 hores següents col·lectivament. El que s'ha mantingut respecte de la metodologia original ha sigut principalment, la idea extreta del manifest de desenvolupament àgil de software que prioritza aportar valor amb software que funcioni per sobre de documentació extensa. També s'ha aprofitat de SCRUM la idea de tenir un product backlog amb les tasques a realitzar en cadascun dels sprints en els quals s'ha dividit el procés d'implementació, el planning de cada sprint abans de començar-lo i la revisió, en forma de reunions amb els directors, un cop acabat cada sprint.

Cadascun dels sprints realitzats ha correspost amb els casos d'ús especificats, excepte l'últim, que ha servit per a fer una revisió general de funcionalitats i disseny, prèvia a la fase de proves del sistema.

Per últim, i com que la fase de proves ha acabat abans del que s'havia planejat, s'ha dedicat temps restant a revisió d'errors que s'han anat trobant durant la fase de proves.

9.3.3 Desplegament

De cara a aquesta fase s'ha decidit utilitzar Heroku, i, en lloc d'esperar que el projecte estigués acabat, s'ha pres la decisió de desplegar el repositori durant la fase d'implementació, per tal de poder donar als directors la possibilitat de veure l'estat del projecte sense haver d'instal·lar cap de les eines de desenvolupament als seus equips.

A continuació s'inclou el diagrama de desplegament corresponent al procés realitzat en el projecte:

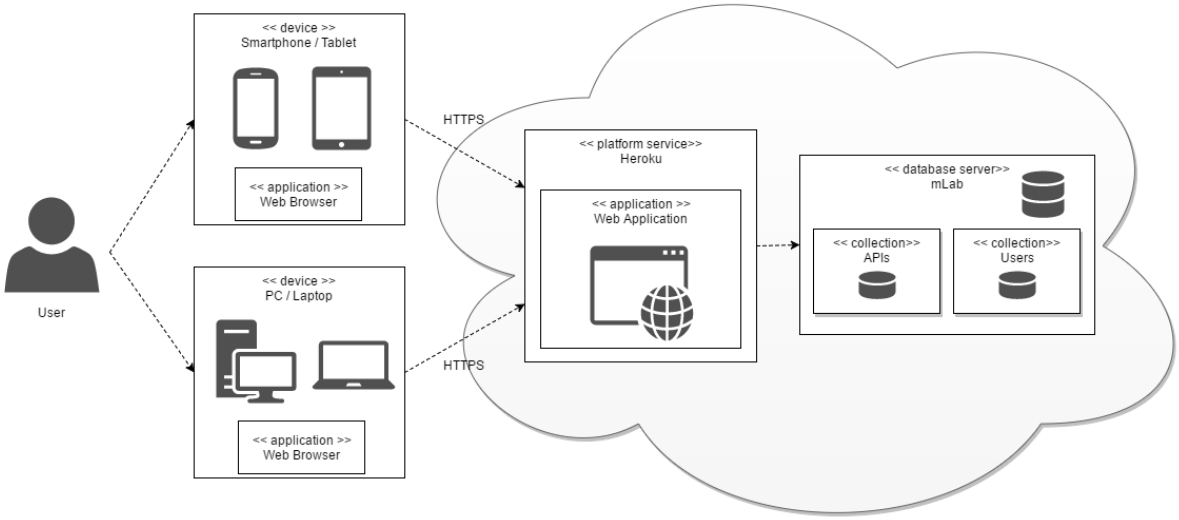


Figura 17: Diagrama de desplegament

Com es pot veure, qualsevol usuari pot connectar-se a l'aplicació, allotjada a Heroku, des del navegador del seu dispositiu. Per darrere, l'aplicació es comunica amb la base de dades allotjada a mLab mitjançant el plugin per a MongoDB que mLab, la plataforma de hosting per a bases de dades MongoDB, ofereix als usuaris d'Heroku.

10. Proves

Un cop acabada la implementació del sistema s'han provat per una banda, les funcionalitats de l'API, i s'han avaluat els requisits no funcionals del sistema, per l'altra.

10.1 Proves de funcionalitats

L'API s'ha provat utilitzant Mocha i Chai, explicats a l'apartat anterior. Per a realitzar les proves, s'ha decidit fer un seguit de proves seqüencials que fan servir totes les funcions de l'API, cridant a aquelles que poden provocar diversos tipus d'errors més d'un cop per comprovar que tot és correcte.

Els tests realitzats són els següents:

- Obtenir totes les APIs: S'espera que el servidor retorni un codi 200 amb un conjunt d'APIs de mida 0, ja que el test s'està realitzant sense haver inserit res a la base de dades.
- Registrar un usuari: Per a aquest test hi ha dues proves:
    - En la primera, s'envia com a body de la crida un nom d'usuari i una contrasenya, i s'espera que el servidor retorni un codi 201 amb un objecte 'user' que tingui com a username el que es passa en el body de la crida i que no tingui nom, ja que en el body de la crida feta per a aquest test només s'hi passa nom d'usuari i contrasenya.
    - En la segona, s'intenta registrar el mateix usuari, i s'espera que el servidor retorni un codi 400 amb un missatge que indiqui que ja hi ha un usuari amb aquest nom d'usuari.
- Iniciar sessió: Per a aquest test hi ha tres proves:
    - En la primera, s'inicia sessió amb l'usuari que s'ha registrat al test anterior, i s'espera que el servidor retorni un codi 200 amb el token que caldrà per autenticar l'usuari en les crides que es faran més endavant i que estan restringides a usuaris que hagin iniciat sessió.
    - En la segona, s'intenta iniciar sessió amb un usuari que no és a la base de dades, i s'espera que el servidor retorni un codi 400 amb un missatge que indiqui que no existeix cap usuari amb aquest nom d'usuari.
    - En la tercera, s'intenta iniciar sessió amb l'usuari que s'ha registrat prèviament, però passant una contrasenya incorrecta, i s'espera que el servidor retorni un codi 400 amb un missatge que indiqui que la contrasenya és incorrecta.
- Afegir una API: Per a aquest test hi ha dues proves:

    - En la primera, s'envia la informació mínima per tal d'afegir una API al sistema (nom, URL de test per NFR i nom d'usuari que l'afegeix) més un vector de categories, que s'aprofitarà en un dels tests següents, i s'espera que el servidor retorni un codi 201 amb un objecte 'api' que sigui igual al slug de l'API el nom de la qual s'ha enviat en el body de la crida.
    - En la segona, s'intenta afegir la mateixa API, i s'espera que el servidor retorni un codi 400 amb un missatge que indiqui que ja hi ha una API amb aquest nom.
- Obtenir una API: S'espera que, en demanar l'API creada en el test anterior, el servidor retorni un codi 200 amb un objecte 'api' que tingui els atributs que s'han proporcionat a l'API en crear-la.
- Obtenir totes les APIs: S'espera que el servidor retorni un codi 200 amb un conjunt d'APIs de mida 1, ja que a la base de dades només hi ha d'haver l'API que s'ha afegit en un dels tests anteriors.
- Obtenir totes les categories: S'espera que el servidor retorni un codi 200 amb un conjunt de categories de mida 2, corresponent a les dues categories afegides a l'API que s'ha inserit en un dels tests anteriors.
- Editar la informació d'una API: Per a aquest test s'han definit quatre proves.
    - En la primera s'afegeix una descripció a l'API que s'ha inserit prèviament, i s'espera que el servidor retorni un codi 200 amb un missatge que digui que la informació de l'API ha sigut actualitzada correctament.
    - En la segona es comprova que la descripció d'aquesta API s'hagi actualitzat correctament, i per tant, s'espera que, en obtenir la informació de l'API, el servidor retorni un codi 200 amb un objecte 'api' que tingui com a descripció la indicada en la primera prova.
    - En la tercera s'afegeix un endpoint a l'API que s'ha inserit prèviament, i s'espera que el servidor retorni un codi 200 amb un missatge que digui que la informació de l'API ha sigut actualitzada correctament.
    - En la quarta s'espera que, en obtenir la informació de l'API, el servidor retorni un codi 200 amb un objecte 'api' que tingui un conjunt d'endpoints de mida 1, i que a la posició 0 d'aquest conjunt hi hagi l'endpoint afegit en la prova anterior.
- Esborrar una API: Per a aquest test s'han definit dues proves:
    - En la primera es fa una petició al servidor per esborrar l'única API que s'ha afegit fins ara, i s'espera que el servidor retorni un codi 200 amb un missatge que digui que l'API ha sigut esborrada amb èxit.

    - En la segona es vol comprovar que l'API ha sigut esborrada, i per tant es demana al servidor que retorni el conjunt de totes les APIs, que s'espera que sigui 0, ja que s'ha esborrat l'única que hi havia fins ara.
- Obtenir un usuari: S'espera que el servidor retorni un codi 200 amb un objecte 'user' que tingui com a username el que s'ha definit en el test corresponent al registre d'un usuari.
- Editar la informació d'un usuari: Per a aquest test s'han definit quatre proves:
    - En la primera, s'edita la informació de l'únic usuari del sistema afegint-li nom, i s'espera que el servidor retorni un codi 200 amb un missatge que indiqui que l'actualització s'ha realitzat.
    - En la segona, es demana la informació de l'usuari editat per tal de comprovar que efectivament ara té una propietat 'name' amb el valor que s'ha indicat a la prova anterior.
    - En la tercera es canvia la contrasenya de l'usuari, i per fer-ho s'envia la contrasenya actual i la nova contrasenya. S'espera que el servidor contesti amb un codi 200 i un missatge que digui que la informació de l'usuari ha sigut editada correctament.
    - En la quarta prova es pretén repetir la prova anterior per veure que, com que la contrasenya que es passa com a actual de l'usuari ja no coincideix amb la que hi ha guardada a la base de dades, el servidor retorna un codi 400 amb un missatge que indica que la contrasenya guardada no coincideix amb l'enviada.
- Esborrar un usuari: En aquesta prova es vol veure que, en fer una petició al servidor per a esborrar l'únic usuari que hi ha, aquest retorna un codi 200 amb un missatge que indica que l'usuari ha sigut esborrat exitosament.

A continuació s'inclou una captura de pantalla amb el resultat dels tests fets:

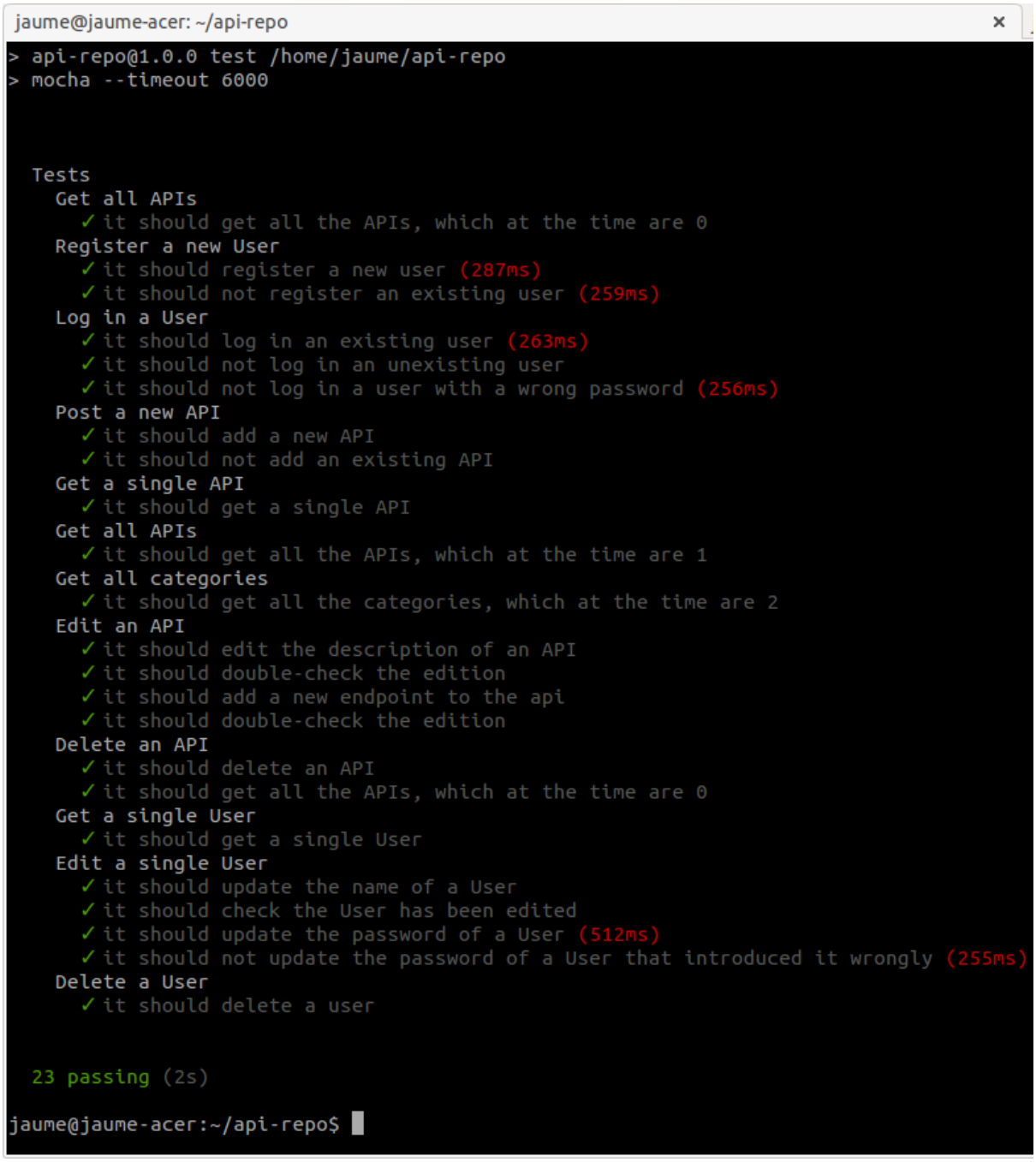


Figura 18: Resultat de les proves de l'API REST

10.2 Proves de requisits no funcionals

Pel que fa als requisits no funcionals, s'ha comprovat el compliment dels seus criteris de satisfacció de forma diferent segons si es tractava de criteris quantitatius o qualitatius.

Els requisits que compten amb criteris quantitatius són els referents a atractiu, estil, facilitat d'ús, claredat de llenguatge i adaptabilitat, i per a tots ells, el criteri de satisfacció és que, com a mínim, un 75% dels usuaris que valorin la qualitat avaluada, ho facin positivament.

Per a avaluar aquests requisits, s'ha elaborat un qüestionari en línia que s'ha distribuït tant a gent coneixedora de l'entorn com a gent que no hi està familiaritzada, amb l'intenció d'aconseguir una mostra heterogènia per a la prova.

Degut a una inesperada falta de temps, el qüestionari només ha pogut estar actiu durant 24 hores, aconseguint un total de 21 respostes.

De les 21 persones que han contestat, n'hi ha 11 que coneixen el concepte d'API, així com el seu funcionament, ja que són estudiants d'enginyeria informàtica, i 10 que no el coneixen, ja que tenen perfils tècnics no relacionats amb l'àmbit de la informàtica o, directament, perfils no tècnics. Pel que fa a la seva edat, la de 19 d'ells està compresa entre els 18 i els 25 anys, i la dels 2 restants, entre els 25 i els 35 anys.

A continuació es mostra el resultat de l'avaluació dels cinc requisits que compten amb criteris quantitatius:

- NFR #13 (Atractiu)

Figura 19: NFR#13 - Atractiu

- NFR #14 (Estil)

Figura 20: NFR#14 - Estil

- **NFR #15 (Facilitat d’ús)**

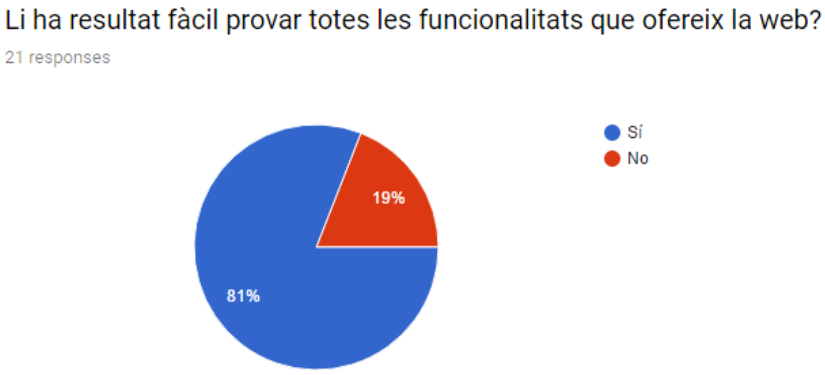


Figura 21: NFR#15 - Facilitat d'ús

- NFR #16 (Claredat en el llenguatge)

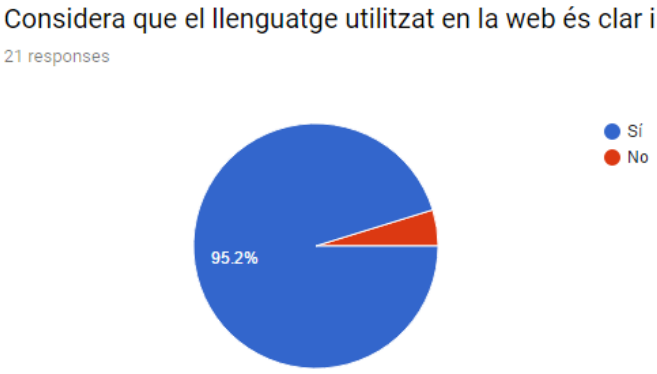


Figura 22: NFR#16 - Claredat en el llenguatge

- NFR #20 (Adaptabilitat)

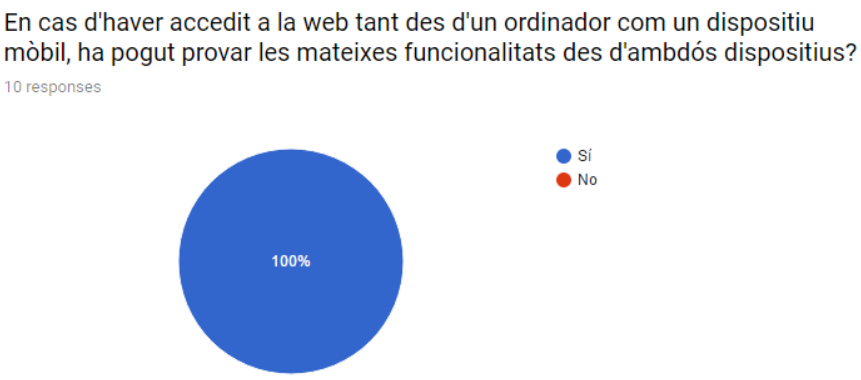


Figura 23: NFR#20 - Adaptabilitat

Com es pot veure, tots els requisits no funcionals s'han assolit correctament, però cal tenir en compte que en el cas de l'atractiu el criteri s'ha complert per molt poc, i que, per tant, l'aspecte visual de la web és un tema important a tenir en compte en el moment que es decideixi millorar les característiques de l'aplicació. De la mateixa forma, també es pot veure que amb un 81%, la facilitat d'ús el segon requisit menys valorat, així que s'hauria de tenir en compte igual que passa amb l'atractiu de cara a futures millores.

Pel que fa als criteris qualitatius, s'han avaluat de la forma següent:

- NFR #17 (Velocitat): El criteri de satisfacció d'aquest requisit és que no hi hagi cap crida a la API interna del sistema que trigui més de 2 segons en retornar una resposta. Com es pot veure en la captura de la realització dels tests de l'apartat 10.1, no hi ha cap crida que superi aquest valor.
- NFR #18 (Disponibilitat): El criteri de satisfacció d'aquest requisit és que el sistema tingui un mínim d'un 95% de temps en línia. L'encarregat de proporcionar aquesta disponibilitat és Heroku, el servidor on s'allotja l'aplicació. Com es pot veure en la pàgina que Heroku proporciona per a veure el seu estat i l'històric de temps en línia [34], aquest és generalment de més del 99%, per tant, es pot assegurar que es compleix.
- NFR #19 (Extensibilitat): El criteri de satisfacció d'aquest requisit es considera complert ja que s'ha tingut present durant totes les fases del projecte i s'ha treballat utilitzant patrons arquitectònics, patrons de disseny i tècniques de programació que promouen la modularitat i minimitzen l'acoblament, assolint així que el sistema resultant sigui fàcilment extensible.
- NFR #21 (Integritat): El criteri de satisfacció d'aquest requisit és que tots els formularis de la web tinguin sistemes de control que comprovin la integritat de les dades introduïdes pels usuaris. En els components que representen els controladors de l'aplicació hi ha control d'entrada per a tots els formularis, ja sigui a nivell del mateix controlador o en l'API REST, sempre intentant detectar errors el més aviat possible. És per això que es considera que es compleix aquest requisit.
- NFR #22 (Privacitat): El criteri de satisfacció d'aquest requisit es compleix, ja que no hi ha cap manera de veure dades de cap dels usuaris sense haver-se registrat en el sistema, un cop registrat, les dades mostrades són només les públiques de cada usuari (nom i nom d'usuari), i les contrasenyes de tots els usuaris es guarden encriptades a la base de dades gràcies a utilitzar la llibreria d'encriptació Bcrypt-nodejs, descrita a l'apartat 9.2.7 d'aquest projecte.

11. Conclusions

En aquest apartat s'inclouen la justificació de competències, les conclusions del projecte, les dificultats que s'han trobat durant el desenvolupament del projecte, i la feina que hi hauria per fer en un futur si es decidís ampliar el sistema que s'ha desenvolupat.

11.1. Justificació de competències

11.1.1 Coneixements previs

Per a la realització d'aquest projecte s'aprofiten coneixements de diverses assignatures de l'especialitat d'Enginyeria del Software, indicats a continuació:

- Arquitectura del Software (AS): Tots els coneixements relacionats amb patrons arquitectònics i especificació i disseny de sistemes software. Important per al projecte, ja que el que s'ha implementat és justament un sistema software.
- Aplicacions i Serveis Web (ASW): Tot el referent a protocols i a disseny i implementació d'aplicacions i serveis web. És important per al projecte, ja que el sistema software que s'ha desenvolupat és justament una aplicació web.
- Disseny de Bases de Dades (DBD): Tot i que la base de dades que s'ha dissenyat pel projecte no és gaire complexa, sí que n'hi ha d'haver una per tal d'emmagatzemar la informació de les APIs del repositori, per tant, s'aprofiten els coneixements apresos en aquesta assignatura, almenys parcialment.
- Enginyeria de Requisits (ER): D'aquí s'aprofita tot el que té a veure amb l'anàlisi de requisits, la planificació del projecte (objectiu i context del sistema) i part del desenvolupament (casos d'ús, principalment), que coincideix amb diverses de les tasques que s'han realitzat durant el desenvolupament del sistema software.
- Gestió de Projectes de Software (GPS): D'aquesta assignatura es poden aprofitar els conceptes teòrics de tot el referent a la gestió de projectes de software i el treball amb metodologies de desenvolupament àgils.
- Projecte d'Enginyeria del Software (PES): Com en el cas de GPS, d'aquesta assignatura també s'aprofiten conceptes de gestió de projectes de software, tot i que a un nivell més pràctic.

11.1.2 Justificació de competències

El projecte s'adequa a les característiques de l'especialitat principalment per ser una part d'un projecte del grup de recerca del departament d'Enginyeria del Software i Sistemes d'Informació de la Universitat Politècnica de Catalunya.

En concret, es pot dir que el projecte és adequat per a l'especialitat, ja que es tracta del desenvolupament d'un sistema software, concretament una aplicació web, que s'utilitza com a repositori d'APIs, i que ha sigut especificat, dissenyat i implementat, tasques que pertanyen a les activitats realitzades a l'especialitat.

A més, durant la realització del projecte s'han tingut en compte diversos rols que es tenen presents a l'especialitat a l'hora de dur a terme projectes, com poden ser el de director de projecte, analista, enginyer de software o el de tester.

Per últim, un altre factor que fa que el projecte s'adeqüi a les característiques de l'especialitat és que s'hi utilitzen molts dels coneixements apresos en assignatures d'aquesta especialitat, com es pot veure en l'apartat anterior d'aquest document.

11.1.3 Competències tècniques

Les competències tècniques que es tenen en compte al projecte són les següents:

- CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics. [Una mica]

Es té només en compte una mica, ja que el sistema software desenvolupat no és extremadament complex ni crític. Es pot comprovar el seu assoliment, ja que el repositori d'APIs ha sigut desenvolupat amb èxit.

- CES1.2: Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles. [En profunditat]

El principal objectiu del projecte és desenvolupar un repositori d'APIs on mostrar i mantenir informació de les APIs que s'hi registrin, per tal de facilitar l'adopció d'aquestes APIs per part dels clients potencials, per tant, es pot dir que es tracta un problema d'integració. L'assoliment d'aquesta competència depèn del feedback positiu dels usuaris que gràcies al repositori poden trobar informació sobre APIs més fàcilment.

- CES1.4: Desenvolupar, mantenir i avaluar serveis i aplicacions distribuïdes amb suport de xarxa. [En profunditat]

Es tracta en profunditat, ja que aquest sistema software és concretament una aplicació web distribuïda en xarxa, que encaixa, per tant, amb la descripció de la competència. De nou, es pot comprovar que s'ha assolit aquesta competència, ja que s'ha implementat el repositori d'APIs en forma d'aplicació en xarxa.

- CES1.5: Especificar, dissenyar, implementar i avaluar bases de dades. [Bastant]

No es considera que es tracti aquesta competència en profunditat, ja que la base de dades que conté informació sobre les APIs que hi ha al repositori és relativament senzilla, però tot i així, sí que és cert que s'ha hagut de dissenyar de forma correcta abans de ser implementada. Es pot veure l'assoliment d'aquesta competència veient que efectivament s'ha implementat una base de dades per al sistema.

- CES1.6: Administrar bases de dades [Una mica]

Es considera que aquesta competència només es té una mica en compte, ja que el gruix de feina pel que fa a base de dades recau en dissenyar-la i implementar-la, no en administrar-la. Efectivament, un cop finalitzat el projecte es pot afirmar que les tasques d'administració de base de dades tenen una càrrega molt menor en comparació amb el que ha sigut el seu disseny i la seva implementació.

- CES1.7: Controlar la qualitat i dissenyar proves en la producció de software. [Bastant]

Aquesta competència s'assoleix gràcies al conjunt de proves que s'han elaborat per tal de comprovar el funcionament del sistema implementat.

- CES1.9: Demostrar comprensió en la gestió i govern dels sistemes software. [Una mica]

Com passa en el cas de la base de dades, la gestió no és la principal tasca a realitzar pel que fa al sistema software, per tant, es considera que la competència només es té una mica en compte.

- CES2.1: Definir i gestionar els requisits d'un sistema software. [En profunditat]

Aquesta competència es treballa en profunditat en el desenvolupament d'aquest sistema, ja que la part de definició dels requisits del sistema ha sigut de les més importants del procés.

- CES2.2: Dissenyar solucions apropiades en un o més dominis d'aplicació, utilitzant mètodes d'enginyeria del software que integrin aspectes ètics, socials, legals i econòmics. [Bastant]

Aquesta competència s'ha tingut bastant en compte, ja que durant la fase de planificació s'han tractat aspectes socials i econòmics en l'àmbit de la sostenibilitat.

11.2 Conclusions

Un cop acabat el projecte es pot assegurar que s'han complert els objectius plantejats, ja que s'ha desenvolupat un repositori d'APIs amb les funcionalitats necessàries per mostrar i mantenir informació completa i acurada, incloent-hi requisits no funcionals, de les API que s'hi registren.

El desenvolupament d'aquest projecte ha contribuït en l'adquisició i la consolidació d'una sèrie de coneixements molt importants per a un enginyer informàtic, per una banda, i ha ajudat a reforçar-ne molts dels obtinguts durant el grau, per l'altra.

Primerament, m'ha permès treballar aspectes de gestió de projectes com són la planificació temporal, l'estimació de costos o la sostenibilitat, en els quals gairebé no havia treballat en cap ocasió prèviament, i m'ha ajudat a familiaritzar-me amb diversos dels rols que participen en el desenvolupament d'un projecte de software, com el de project manager o el d'analista de requisits, ja que en tractar-se d'un projecte individual, he hagut de dur-los a terme variant segons l'etapa del desenvolupament en la que em trobés en cada moment. De la mateixa forma, també he repassat continguts treballats en moltes de les assignatures del grau, i més concretament de l'especialitat d'Enginyeria del Software com l'anàlisi de requisits o el disseny d'aplicacions web, entre d'altres.

A més, realitzar aquest projecte també m'ha permès, per una banda, consolidar els meus coneixements de desenvolupament d'aplicacions web amb MEAN Stack i les tecnologies que s'hi relacionen, i per l'altra, treballar amb eines que no havia fet servir fins ara per a realitzar proves automatitzades, com Mocha o Chai, que han resultat ser molt útils per a aquesta fase del desenvolupament del projecte, ja no només per aquest projecte en concret sinó per a qualsevol que pugui desenvolupar en un futur.

En definitiva, l'aplicació creada permet mostrar informació sobre APIs de forma que aquells desenvolupadors que ho decideixin hi puguin mostrar informació sobre les seves APIs, i aquells que, al seu torn, busquin informació sobre altres APIs, també puguin fer-ho en el mateix lloc.

11.3 Dificultats

La principal dificultat que m'he trobat durant el desenvolupament ha sigut fer una planificació viable per al projecte, ja que aquest projecte ha sigut el primer que he hagut de desenvolupar íntegrament en solitari.

Tot i això, considero que, després d'haver sigut revisada (com s'ha vist en l'apartat de planificació temporal) la planificació ha esdevingut viable.

Pel que fa a aspectes tècnics, la part del desenvolupament que més dificultats m'ha plantejat ha sigut aquella relacionada amb el front-end de la web, ja que, tot i que és cert que havia treballat algun cop amb les tecnologies utilitzades per al desenvolupament, no ho havia fet mai sobre la part d'HTML i CSS que donen a la web l'aspecte que té, i això m'ha suposat haver-ne d'aprendre de zero per a aquest projecte.

11.4 Futures extensions

Tot i que, com ja s'ha dit, el projecte ha assolit els objectius marcats inicialment, és cert que hi ha funcionalitats que podrien ser millorades en iteracions futures, de la mateixa forma que se n'hi podrien afegir de noves.

El primer que considero que podria millorar és el disseny de les vistes de la pàgina, ja que com he dit, aquesta ha sigut la meva gran dificultat durant el projecte, i a causa de la limitació temporal del mateix no he pogut dedicar-li tant temps com m'hauria agradat per tal de fer-ne un de molt més cuidat.

Pel que fa a possibles funcionalitats que s'hi podrien afegir, una d'elles podria ser un panell d'administració, per tal que un usuari administrador pogués gestionar la pàgina web en cas que fos necessari.

Una altra funcionalitat que s'hi podria afegir seria la de calcular i/o mostrar més requisits no funcionals de les APIs, ja que per al projecte només se n'han tingut en compte tres. Això es podria fer de, com a mínim dues formes: la primera seria decidir quins més es tenen en compte, i la segona, donar als usuaris que introdueixin informació sobre les seves APIs la llibertat d'especificar quins requisits no funcionals volen que es tinguin en compte, així com la forma de calcular-los, per tal que el sistema fos capaç de fer-ho si fes falta, ja que també podrien ser dades estàtiques que només haurien de mostrar-se com a informació extra de les APIs.

Annexos

Annex 1: API REST

Les crides marcades amb (*) són aquelles que només estan disponibles per a usuaris registrats.

GET /api/apis

Descripció: Crida per a obtenir totes les APIs guardades.

Ruta: /api/apis

Mètode: GET

Resultat:

- Si no es produeix cap error:
  - Status: 200 OK
  - JSON:

```
{
  "success": true,
  "apis":
  [
    {
      "name": "Name1",
      "slug": "name1",
      "description": "Description1",
      "tags": ["tag1", "tag2"]
    },
    {
      "name": "Name2",
      "slug": "name2",
      "description": "Description2",
      "tags": ["tag1", "tag3"]
    }
  ]
}
```

- Si es produeix algun error:
  - Status: 404 Not found
  - JSON:

```
{
```

```
  "success": false,
  "message": "Error message"
}
```

POST /api/apis (*)

Descripció: Crida per a afegir un nou usuari a la base de dades.

Ruta: /api/apis

Mètode: POST

Body:

- name: Nom de l'API que es vol afegir.
- description: Descripció de l'API que es vol afegir. (opcional)
- tags: Categories associades a l'API que es vol afegir. (opcional)
- portal: URL base de l'API que es vol afegir. (opcional)
- testUrl: URL triada per al càlcul dels requisits no funcionals de l'API.
- endpoints: Endpoints amb els que conta l'API. (opcional)
- protocol: Protocol de comunicació de l'API. (opcional)
- reqFormat: Format que accepta l'API per a les crides que rep. (opcional)
- resFormat: Format que utilitza l'API en les respostes que proporciona. (opcional)
- libs: Llibreries externes que fan servir aquesta API. (opcional)
- devs: Desenvolupadors de l'API. (opcional)
- pricing: Preu que s'ha de pagar per a fer servir l'API (en cas que no sigui gratuit). (opcional)

Resultat:

- Si no es produeix cap error:
  - Status: 201 Created
  - JSON:

    ```
    {
      "success": true,
      "api": "api-slug"
    }
    ```
- Si es produeix algun altre error:
  - Status: 400 Bad request
  - JSON:

    ```
    {
    ```

```
  "success": false,
  "message": "Error message"
}
```

GET /api/tags

Descripció: Crida per a obtenir les categories de totes les APIs guardades.

Ruta: /api/tags

Mètode: GET

Resultat:

- Si no es produeix cap error:
  - Status: 200 OK
  - JSON:

```
{
  "success": true,
  "tags": ["tag1", "tag2", "tag3"]
}
```

- Si es produeix algun error:
  - Status: 404 Not found
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

GET /api/apis/:slug

Descripció: Crida per a obtenir l'API identificada per 'slug'.

Ruta: /api/tags/:slug

Mètode: GET

Paràmetres:

- slug: Slug identificatiu de l'API que es vol retornar.

Resultat:

- Si no es produeix cap error:
  - Status: 200 OK

- JSON:


```
{
  "success": true,
  "api": {
    "name": "Name1",
    "slug": "name1",
    "description": "Description 1",
    "tags": ["tag1", "tag2", "tag3"],
    "portal": "http://localhost:3000",
    "testUrl": "http://localhost:3000/api/test",
    "endpoints":
    [
      {
        "name": "Endpoint1",
        "description": "Endpoint 1 description."
      }
    ],
    "protocol": "REST",
    "reqFormat": "JSON",
    "resFormat": "JSON",
    "libs":
    [
      {
        "name": "Lib1",
        "description": "http://link1url.com"
      }
    ],
    "devs":
    [
      {
        "name": "John Doe",
        "website": "http://itsjohndoe.me"
      }
    ],
    "pricing": ["Free to use"],
    "totalReq": "200",
    "totalRes": "188",
    "correctRes": "175",
    "totalResTime": "600000"
```

```
    }
  }
```

- Si es produeix algun error:
  - Status: 404 Not found
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

PUT /api/apis/:slug (*)

Descripció

Ruta: /api/tags/:slug

Mètode: PUT

Paràmetres:

- slug: Slug identificatiu de l'API que es vol editar.

Body (tots els elements són opcionals):

- description: Nova descripció de l'API.
- tags: Nou llistat de categories associades a l'API.
- portal: Nova URL base de l'API.
- testUrl: Nova URL per al càlcul dels requisits no funcionals.
- endpoints: Nou llistat d'endpoints de la API.
- protocol: Nou protocol de comunicació de l'API.
- reqFormat: Nou format que accepta l'API per a les crides que rep.
- resFormat: Nou format que utilitza l'API en les respostes que proporciona.
- libs: Nou llistat de llibreries externes que fan servir aquesta API.
- devs: Nou llistat de desenvolupadors de l'API.
- pricing: Nou preu que s'ha de pagar per a fer servir l'API (en cas que no sigui gratuit).
- time: Paràmetre intern usat per a sumar-ne el valor al temps de resposta total de l'API.
- correct: Paràmetre intern que indica que s'ha produit una resposta correcta per part de l'API en qüestió en l'última crida feta per al càlcul de requisits no funcionals.

Resultat:

- Si no es produeix cap error:

    - Status: 200 OK
    - JSON:

```
{
  "success": true,
  "message": "API information updated successfully"
}
```

- Si es produeix algun error:
    - Status: 400 Bad request
    - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

DELETE /api/apis/:slug (*)

Descripció: Crida per a esborrar de la base de dades l'API identificada per 'slug'.

Ruta: /api/apis/:slug

Mètode: DELETE

Paràmetres:

- slug: Slug identificatiu de l'API que es vol editar.

Resposta:

- Si l'API s'esborra correctament:
    - Status: 200 OK
    - JSON:

```
{
  "success": true,
  "message": "Successfully deleted"
}
```

- Si es produeix algun error:
    - Status: 400 Bad request
    - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

POST /api/users

Descripció: Crida per a afegir un nou usuari a la base de dades.

Ruta: /api/users

Mètode: POST

Body:

- name: Nom de l'usuari que es vol afegir. (opcional)
- username: Nom d'usuari de l'usuari que es vol afegir.
- password: Contrassenya de l'usuari que es vol afegir.

Resultat:

- Si no es produeix cap error:
  - Status: 201 Created
  - JSON:

```
{
  "success": true,
  "user": {
    "username": "username",
    "name": "John Doe"
  }
}
```

- Si ja hi ha un usuari amb aquest username:
  - Status: 400 Bad Request
  - JSON:

```
{
  "success": false,
  "message": "A user with that username already exists"
}
```

- Si es produeix algun altre error:
  - Status: 404 Not found
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

GET /api/users/:username (*)

Descripció: Crida per a obtenir l'usuari identificat per 'username'.

Ruta: /api/users/:username

Mètode: GET

Paràmetres:

- username: Nom d'usuari de l'usuari que es vol obtenir.

Resultat:

- Si no es produeix cap error:
  - Status: 200 OK
  - JSON:

```
{
  "success": true,
  "user": {
    "username": "username",
    "name": "John Doe"
  }
}
```

- Si es produeix algun error:
  - Status: 404 Not found
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

PUT /api/users/:username (*)

Descripció: Crida per a editar la informació de l'usuari identificat per 'username'.

Ruta: /api/users/:username

Mètode: PUT

Paràmetres:

- username: Nom d'usuari de l'usuari que es vol editar.

Body (tots els elements són opcionals):

- name: Nou nom per a l'usuari.
- oldPassword: Actual contrassenya de l'usuari.
- newPassword: Nova contrassenya per a l'usuari.

Resultat:

- Si no es produeix cap error:
  - Status: 200 OK
  - JSON:

```
{
  "success": true,
  "message": "User updated successfully"
}
```

- Si la contrassenya antiga no coincideix:
  - Status: 400 Bad request
  - JSON:

```
{
  "success": false,
  "message": "Wrong old password"
}
```

- Si es produeix algun altreerror:
  - Status: 400 Bad request
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

DELETE /api/users/:username (*)

Descripció: Crida per a esborrar de la base de dades l'usuari identificat per 'username'.

Ruta: /api/users/:username

Mètode: DELETE

Paràmetres:

- username: Nom d'usuari de l'usuari que es vol esborrar.

Resposta:

- Si l'usuari s'esborra correctament:

    - Status: 200 OK
    - JSON:

```
{
  "success": true,
  "message": "Successfully deleted"
}
```

- Si es produeix algun error:
    - Status: 400 Bad request
    - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

POST /api/authenticate

Descripció: Crida per a autenticar un usuari quan inicia sessió.

Ruta: /api/authenticate

Mètode: POST

Body:

- username: Nom d'usuari de l'usuari que es vol autenticar.
- password: Contrassenya de l'usuari que es vol autenticar.

Resultat:

- Si no es produeix cap error:
    - Status: 200 OK
    - JSON:

```
{
  "success": true,
  "token": "Token value"
}
```

- Si no hi ha cap usuari amb aquest username:
    - Status: 404 Not found
    - JSON:

```
{
  "success": false,
  "message": "Authentication failed. User not found"
```

```
}
```

- Si la contrassenya introduida no és correcta:
  - Status: 403 Forbidden
  - JSON:

```
{
  "success": false,
  "message": "Authentication failed. Wrong password"
}
```

- Si es produeix algun altre error:
  - Status: 400 Bad Request
  - JSON:

```
{
  "success": false,
  "message": "Error message"
}
```

Annex 2: Funcionalitats dels repositoris estudiats

Funcionalitat	ProgrammableWeb	Exicon	Algorithmia	API Harmony	RapidAPI
Notícies	X			X	X
Perfil API	X	X	X	X	X
Perfil usuari	X	X	X		X
Catàleg API	X				
Afegir API	X	X		X	X
Cercador	X	X	X		X
Classificació	X		X	X	X
Seguiment	X				
Newsletter	X				
Prova			X		

Taula 7: Estat de l'art: funcionalitats

- Notícies: Apartat de la web on els usuaris poden penjar articles.
- Perfil API: Pàgina on es mostra tota la informació d'una API.
- Perfil usuari: Pàgina on es mostra tota la informació d'un usuari.
- Catàleg APIs: Llistat de totes les APIs que hi ha al repositori.
- Afegir API: Possibilitat d'afegir informació sobre una API nova al sistema.
- Cercador: Possibilitat de cercar APIs per text.
- Classificació d'APIs: Possibilitat de mostrar només les APIs de certes categories.
- Seguiment: Possibilitat de mostrar interès en una API, com a usuari, a l'estil de les xarxes socials.
- Newsletter: Subscripció setmanal a notificacions via e-mail de notícies del repositori.
- Prova: Possibilitat de fer crides a les APIs des del mateix repositori.

Annex 3: Informació mostrada pels repositoris estudiats

Informació	ProgrammableWeb	Exicon	Algorithmia	API Harmony	RapidAPI
Descripció	X	X	X	X	
Portal	X	X		X	
Endpoints	X			X	X
Categoria	X	X			
Documentació	X			X	
Protocol	X	X			
Request formats	X	X			
Response formats	X	X			
Exemples	X				
Llibreries externes	X			X	
Desenvolupadors	X	X			
Seguidors	X				
Comentaris	X				
Pricing		X	X		
Especificació OpenAPI				X	
Temes a StackOverflow				X	
Usos a Github				X	
Última modificació					X

Taula 8: Estat de l'art: informació proporcionada

- Descripció: Breu introducció on s'explica per a què es fa servir l'API en qüestió.
- Portal: URL corresponent al punt d'entrada de l'API.
- Endpoints: Conjunt d'URLs corresponents als diferents punts d'informació que proporciona l'API.
- Categoria: Categoria sota la qual es classifica l'API en el repositori.
- Documentació: URL amb la documentació (generalment) oficial de l'API.
- Arquitectura / protocol: Arquitectura o protocol amb el qual ha estat implementada l'API.
- Formats de request: Formats que admet l'API en les crides fetes contra ella.
- Formats de response: Formats que l'API usa en les respostes que envia.
- Exemples: Exemples de codi on es fan servir diverses funcionalitats de les APIs.
- Llibreries: Llibreries externes creades per a interactuar amb l'API.
- Desenvolupadors: Perfils dels desenvolupadors de l'API (o links externs si la pàgina no ofereix perfils per a usuaris)
- Seguidors: Perfils dels usuaris que han mostrat el seu interès en l'API.
- Comentaris: Apartat on els usuaris poden dir la seva sobre la API, demanar ajuda amb problemes sorgits durant intents de connexió, etc.
- Pricing: Preu a pagar per a utilitzar l'API.
- Especificació OpenAPI: Document amb l'especificació de l'API en format JSON, que en conté tota la informació que el seu creador hagi decidit incloure-hi.
- Temes a StackOverflow: Llistat amb els temes més populars a StackOverflow on la gent demana ajuda a l'hora d'usar aquesta API.
- Usos a Github: Nombre (i llistat) de projectes allotjats a Github que fan crides a aquesta API.
- Última modificació: Data de l'última modificació de l'API.

Annex 4: Bibliografia

[1] “Web APIs for non-programmers,” [Online]. Available: https://schoolofdata.org/2013/11/18/web-apis-for-non-programmers/. [Accessed 22 February 2017].

[2] «What are Non-Functional Requirements?,» [En línia]. Available: http://reqtest.com/requirements-blog/what-are-non-functional-requirements/. [Últim accés: 26 February 2017].

[3] «API Discovery: 11+ Ways to find APIs,» [En línia]. Available: http://nordicapis.com/api-discovery-11-ways-to-find-apis/. [Últim accés: 27 January 2017].

[4] «ProgrammableWeb,» [En línia]. Available: http://programmableweb.com/. [Últim accés: 27 January 2017].

[5] «Exicon,» [En línia]. Available: https://app.exiconglobal.com/api-dir/. [Últim accés: 27 January 2017].

[6] «Algorithmia,» [En línia]. Available: https://algorithmia.com/algorithms. [Últim accés: 27 January 2017].

[7] «API Harmony,» [En línia]. Available: https://apiharmony-open.mybluemix.net/. [Últim accés: 27 January 2017].

[8] «The OpenAPI-Specification,» [En línia]. Available: https://github.com/OAI/OpenAPI-Specification. [Últim accés: 27 February 2017].

[9] «StackOverFlow,» [En línia]. Available: http://stackoverflow.com/company/about. [Últim accés: 27 February 2017].

[10] «GitHub,» [En línia]. Available: https://github.com/about. [Últim accés: 27 February 2017].

[11] «RapidAPI,» [En línia]. Available: https://rapidapi.com/. [Últim accés: 27 January 2017].

[12] «Bitbucket,» [En línia]. Available: https://www.atlassian.com/software/bitbucket. [Últim accés: 8 June 2017].

[13] «What is a framework?,» [En línia]. Available: http://whatis.techtarget.com/definition/framework. [Últim accés: 3 March 2017].

[14] «What is Scrum?,» [En línia]. Available: https://www.scrum.org/resources/what-is-scrum. [Últim accés: 3 March 2017].

[15] «Glassdoor Job Search,» [En línia]. Available: https://www.glassdoor.com/index.htm. [Últim accés: 8 March 2017].

[16] «Requirements Specification Template,» [En línia]. Available: http://www.volere.co.uk/template.htm. [Últim accés: 16 June 2017].

[17] «What is a use case?,» [En línia]. Available: http://searchsoftwarequality.techtarget.com/definition/use-case. [Últim accés: 16 June 2017].

[18] «MVC Architecture,» [En línia]. Available: https://developer.chrome.com/apps/app_frameworks. [Últim accés: 28 May 2017].

[19] «Introducing JSON,» [En línia]. Available: www.json.org. [Últim accés: 27 May 2017].

[20] «What is HTML?,» [En línia]. Available: http://www.yourhtmlsource.com/starthere/whatishtml.html. [Últim accés: 27 May 2017].

[21] «CSS Introduction,» [En línia]. Available: https://www.w3schools.com/css/css_intro.asp. [Últim accés: 27 May 2017].

[22] «What is JavaScript?,» [En línia]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Últim accés: 28 May 2017].

[23] «What is MongoDB?,» [En línia]. Available: https://www.mongodb.com/what-is-mongodb. [Últim accés: 28 May 2017].

[24] «Express.js: A Server-Side JavaScript Framework,» [En línia]. Available: https://www.upwork.com/hiring/development/express-js-a-server-side-javascript-framework/. [Últim accés: 28 May 2017].

[25] «What is AngularJS?,» [En línia]. Available: https://docs.angularjs.org/guide/introduction. [Últim accés: 28 May 2017].

[26] «Node.js - Introduction,» [En línia]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Últim accés: 28 May 2017].

[27] «What is Heroku?,» [En línia]. Available: https://www.heroku.com/what. [Últim accés: 29 May 2017].

[28] «Bootstrap Introduction,» [En línia]. Available: http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php. [Últim accés: 4 June 2017].

[29] «What is front-end?,» [En línia]. Available: http://whatis.techtarget.com/definition/front-end. [Últim accés: 20 June 2017].

[30] «Mocha,» [En línia]. Available: https://mochajs.org/. [Últim accés: 12 June 2017].

[31] «Chai,» [En línia]. Available: http://chaijs.com/. [Últim accés: 12 June 2017].

[32] «Agile Alliance: Behaviour Driven Development,» [En línia]. Available: https://www.agilealliance.org/glossary/bdd/. [Últim accés: 12 June 2017].

[33] «Agile Alliance: Test Driven Development,» [En línia]. Available: https://www.agilealliance.org/glossary/tdd/. [Últim accés: 12 June 2017].

[34] «Heroku Status,» [En línia]. Available: https://status.heroku.com/. [Últim accés: 8 June 2017].

[35] «Manifesto for Agile Software Development,» [En línia]. Available: http://agilemanifesto.org/. [Últim accés: 6 June 2017].